Peregrine

AssetCenter

Advanced use

Peregrine SYSTEMS

DAC-43-EN08

© Copyright 2003 Peregrine Systems, Inc. or its subsidiaries.

All Rights Reserved.

Information contained in this document is proprietary to Peregrine Systems, Incorporated, and may be used or disclosed only with written permission from Peregrine Systems, Inc. This manual, or any part thereof, may not be reproduced without the prior written permission of Peregrine Systems, Inc. This document refers to numerous products by their trade names. In most, if not all, cases these designations are claimed as Trademarks or Registered Trademarks by their respective companies.

Peregrine Systems ® and AssetCenter ® are trademarks of Peregrine Systems, Inc. or its subsidiaries.

This document and the related software described in this manual are supplied under license or nondisclosure agreement and may be used or copied only in accordance with the terms of the agreement. The information in this document is subject to change without notice and does not represent a commitment on the part of Peregrine Systems, Inc. Contact Peregrine Systems, Inc., Customer Support to verify the date of the latest version of this document.

The names of companies and individuals used in the sample database and in examples in the manuals are fictitious and are intended to illustrate the use of the software. Any resemblance to actual companies or individuals, whether past or present, is purely coincidental.

If you need technical support for this product, or would like to request documentation for a product for which you are licensed, contact Peregrine Systems, Inc. Customer Support by email at support@peregrine.com.

If you have comments or suggestions about this documentation, contact Peregrine Systems, Inc. Technical Publications by email at doc_comments@peregrine.com.

This edition applies to version 4.3 of the licensed program

AssetCenter

Peregrine Systems, Inc.
Worldwide Corporate Campus and Executive Briefing Center
3611 Valley Centre Drive San Diego, CA 92130
Tel 800.638.5231 or 858.481.5000
Fax 858.481.1751
www.peregrine.com

Table of Contents

List of Figures

List of Tables

CHAPTER 1 Itemized lists

An itemized list is a list of values proposed by AssetCenter for populating certain fields (standard fields in a detail screen, or the value of a feature), e.g. Title, Position, Country, Brand.

This enables you to standardize the values in these fields, and facilitates data entry.

The values appear in a "drop-down list". Simply choose the correct value from the list to assign a value to the field.

AssetCenter manages two types of itemized lists:

- Free itemized lists
- System itemized lists

Free itemized lists

The AssetCenter administrator can access free itemized lists using the Administration/ Itemized lists menu item in AssetCenter.

There are two types of free itemized lists:

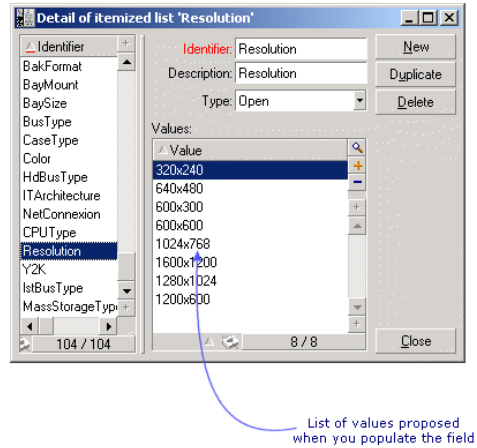
- Itemized lists that you create yourself. You can link them to features, but you cannot attach them to fields: Only the software can assign an itemized list to a field.
- Lists that are linked to database fields. These lists may also be attached to features. If you delete an itemized list of this type, or modify its name, AssetCenter will recreate the itemized list (with no associated values) using its original name, as soon as the itemized list is required to create a record containing a field which should be linked to that list.

Values of itemized lists

The list of "values" appearing in the detail of an itemized list contains the values that will be proposed when the user populates a field associated with the itemized list.

The administrator can delete, modify or add values using the −, and + buttons located to the right of the list.

Figure 1.1. Itemized list - window



Open itemized lists

The **Type** field (SQL name: seType) field in the detail of these itemized lists is set to **Open**.

AssetCenter users can enter other values than those proposed in the list.

If a user enters a new value, it is added to the list of values in the itemized list (the list of values is shared by all users). A message requests the user to confirm the creation.

Closed itemized lists

The **Type** field (SQL name: seType) of the detail for theses itemized lists is set to **Closed**.

AssetCenter users enter only values included in this list.

Note:

After clicking **Create** or **Modify** in the itemized lists management window, any modifications that the administrator defines for itemized lists are stored in the database. Modifications made to itemized list are only taken into account at the client level after the next connection to the database following the modification.

System itemized lists

The list of values in a system itemized list is defined by AssetCenter. It cannot be customized by the administrator or a user.

These itemized lists cannot be edited from the **Administration/ Itemized lists** menu item.

Values in system itemized lists

The values displayed are different from the values stored in the database.

In the database, these values are stored as numbers.

Example of the **Assignment** field (SQL name: seAssignment) in the asset detail:

Table 1.1. Values in system itemized lists

Value stored in the database	Value displayed
0	**In used**
1	**In stock**
2	**Retired asset**
3	**Awaiting delivery**

There are several ways to access the values in a system itemized list:

- Using the context-sensitive help for the field populated by the system itemized list.
- From AssetCenter Database Administrator.
- Using the **database.txt** file describing the database structure.

CHAPTER 2 Historization

Any modifications made to field values and table links in the database can be recorded. Every time you create, modify or delete a value in a field for which the history is kept, AssetCenter creates a history line in the History tab of that screen.

In order to do this you need to specify that "history" is kept for the field or link. To do this:

1 From the shortcut menu, select **Configure object**.

2 Go to the **General** tab in the configuration screen.

3 Set the **History** field to **Yes**.

4 Click **OK** to confirm.

Note:

Any modifications concerning history are saved in the database as soon as you click **OK** in the database customization window. You can also define whether history is kept for a field or a link using AssetCenter Database Administrator.

When history is kept for a field or a link, it is available for all users of AssetCenter.

As soon as history is kept for at least one field or link, a **History** tab appears in the record detail window for this table. "History lines" are kept here, which describe in detail any modifications that have been made to a field or a link.

History lines contain several pieces of information:

- **Modified on** (SQL name: dtLastModif): Date on which the modification was carried out.
- **Author** (SQL name: Author): Person who performed the modification (login, name and first name).
- **Field** (SQL name: Field): Name of the field that has been modified (short description).
- **Previous value** (SQL name: PreviousVal): Previous value of the modified field (except for "comment" type fields).
- **New value** (SQL name: NewVal): New value of the modified field (except for "comment" type fields). By default, this field does not appear in the list. To display it, right-click the list then select **Configure list**.

  Note:

  If you are importing a database from an older version of AssetCenter, the **New value** field will be empty for history lines.

- **Previous comment** (SQL name: memPreviousCmt): Previous value of "comment" type fields. "Comment" type fields are not handled in the same way as other fields, since they are stored differently in the database (max. size: 32767 characters).

Depending on the case, AssetCenter functions differently:

Record creation

Record creations are recorded if you have asked AssetCenter to track history of all modifications made to the identification field corresponding to the primary key of the table).

AssetCenter records:

- **Modified on**: Creation date
- **Author**: Author of the creation
- **Field**: "Creation"
- **Previous value**: "Creation"

Modification of a field in the table or a 1 link (for example: User of an asset)

AssetCenter records:

- **Modified on**: Date on which the modification was made.
- **Author**: Author of the modification.
- **Field**: Name of field modified.
- **Previous value**: Previous value of the modified field.
- **New value**: New value of the modified field.

Addition of a n link to another table (for example: Assets covered by a contract)

AssetCenter records:

- **Modified on**: Date on which the addition was made.
- **Author**: Author of the addition.
- **Field**: Name of link.
- **Previous value**: References of the linked record that has been added.
- **New value**: New value of the modified link.

Deletion of a n link to another table

AssetCenter records:

- **Modified on**: Date on which the deletion was made.
- **Author**: Author of the deletion.
- **Field**: References of the linked record that has been deleted.
- **Previous value**: References of the linked record that has been deleted.
- **New value**: New value of the modified link (empty).

Modification of a n link to another table

AssetCenter does not record modifications made to a link. To keep a trace, you need to delete the obsolete link, then add the new one.

Keeping history of features associated with records

In AssetCenter, you can keep history of features, just like for any other field in the database. Feature history concerns:

- Adding features.
- Removing features.
- Changing the value of a feature.

Several types of actions are kept in history:

Adding a feature

Additions of new features are recorded if the **Keep history** (SQL name: seKeepHistory) field for the feature is set to **Yes** and if the **Keep history even when creating main record** (SQL name: bCreationHistory) box is checked.

AssetCenter records:

- **Modified on** (SQL name: dtLastModif): date when the feature was added.
- **Author** (SQL name: Author): the person who added the feature.
- **Previous value**: "Creation".
- **Field**: SQL name of the feature.

Deleting a feature

Deletions of features are recorded if the **Keep history** field for the feature is set to **Yes**.

AssetCenter records:

- **Modified on**: date when the feature was deleted.
- **Author**: the person who deleted the feature.
- **Field**: SQL name of the feature.
- **Previous value**: " Remove feature ("Value of feature")".

- **New value**: New value of feature (empty).

Modifying the value of a feature

Modifications of features are recorded if the **Keep history** field (SQL name: seKeepHistory) for the feature is set to **Yes**.

AssetCenter records:

- **Modified on**: Date when the feature was modified.
- **Author**: The person who modified the feature.
- **Field**: SQL name of the feature.
- **Previous value**: Previous value of the feature.
- **New value**: New value of the modified feature.

Warning:

If you delete a record, all history lines are also deleted, either straightaway or via AssetCenter Server.

Creating, deleting or modifying history lines

It is not possible to keep history of the creation of history lines.

Creating history lines

To trigger the creation of history lines for a feature, you must set the **Keep history** field to **Yes**. To do this, select the **Parameters** tab in the feature details and click the button opposite the parameters line.

AssetCenter displays the screen with the details of the feature's parameters. The **Keep history** field is located in the **Constraints** tab in this screen.

When this field is set to **Yes**, AssetCenter automatically creates history lines for this feature. History lines may be viewed in the **History** tab of the table associated with this feature.

Warning:

If you delete a record, all history lines are also deleted, either at the time of the deletion, or via AssetCenter Server. You cannot keep history of the creation of histories.

CHAPTER 3 AQL queries

This chapter explains how to write queries in AQL:

Presentation

This section presents the AQL language and lists the places where you might have to use queries:

AQL

AQL ("Advanced Query Language") is the querying language used by AssetCenter to access the AssetCenter database. It is comparable to SQL. Queries written in AQL are automatically translated into the corresponding SQL language of the database engine used.

Note:

It is useful to be well acquainted with SQL and to have good background knowledge of databases before using AQL directly.

Usefulness of AQL language

AQL language is better suited to querying the AssetCenter database than SQL for the following reasons:

Independence from the database engine

The various database engines supported by AssetCenter all use differing versions of SQL that are incompatible with each other. AQL is independent of the database engine used.

As a consequence, if you migrate to another database engine, your queries written in AQL will work the same.

For example, AQL uses the same set of functions, regardless of the database engine used.

Thus, the **Substring** function in AQL is equivalent to **Substr** under SQL Oracle for WorkGroups and **Substring** under Microsoft SQL Server SQL.

Generation of optimized SQL code

AQL generates SQL code optimized according to the database engine used.

This makes a big difference when using indexes. For example, when searching the full names of models by forcing the use of the indexes **ID of the model**(**Model_lModelId**) and **Full name** (**FullName**) you would write the following in AQL:

```
SELECT FIRST_ROWS lModelId, FullName FROM amModel
```

The SQL code generated will differ according to the DBMS used and will be optimized for this. Thus, the SQL Oracle code will be:

```
SELECT /*+ FIRST_ROWS INDEX_ASC(M1 Model_lModelId) */ M1.lModelId, M1.Ful
lName FROM amModel M1
```

The Microsoft SQL Server or Sybase SQL Server code will be:

```
SELECT M1.lModelId, M1.FullName FROM amModel M1 ORDER BY M1.lModelId
```

The IBM DB2 code will be:

```
SELECT lModelId, FullName FROM amModel OPTIMIZE FOR 100 ROWS
```

Simplified access to the AssetCenter database

AQL simplifies the handling of links and joins. This greatly facilitates access to the database when writing queries as compared to using SQL directly.

In addition, AQL simplifies access to features, allowing you to use them as direct fields in their related tables.

AQL also facilitates the utilization of calculated fields.

Specifics of AQL compared to SQL

AQL does not support DDL ("Data Definition Language") orders.

AQL comprises extensions that simplify the handling of joins, features and calculated fields.

Warning:

You should never write to the AssetCenter database directly using SQL statements.

Queries in AssetCenter

Queries enable you to combine criteria concerning information in a table or in linked tables.

You can use queries to:

- Create filters on record lists. In this case, the queries are, in general, simple and based on the "Where" clause.
- Define views.
- Define export conditions in the export module.
- Create Crystal Reports.
- Create wizards.
- When you use the AssetCenter API.
- If AssetCenter is used as a DDE server.

AQL ("Advanced Query Language") is AssetCenter's internal querying language intended to simply access to the AssetCenter database.

AssetCenter includes an editor that enables you to draw up queries in AQL:

- Either indirectly by using the graphic interface.
- Or by writing the query directly in AQL.

Warning:

In order to be illustrative of the capabilities of AQL, the examples given later use all of the AQL syntax. The SELECT, WHERE, and FROM clauses, in particular, are explicitly shown. Certain functions, such as query filters (where the user only defines the WHERE clause in the AQL query) or the expression builder greatly simplify the creation of queries for the user (certain clauses are not visible). These examples cannot be used for these functions.

Recommendations for writing AQL queries

We recommend reading this section before starting to write queries in AQL.

This section deals with:

- Notation specific to AQL.
- The specificities of AQL and the AssetCenter database that have an effect on the optimal design of queries.

The sections entitled AQL syntax [page 43] and AQL function reference [page 56] complement this section.

Warning:

Queries written in AQL use the SQL names ("SQLName") of fields, links and table in the database. Refer to the **database.txt** file that describes the structure of the database and which includes an exhaustive list of these names.

This file is located in the following folder: **[AssetCenter installation folder]/doc/infos**

Presentation of AQL joins

Definition

A join consolidates multiple data tables in a single query.

AQL joins

AssetCenter's database description, besides defining the tables and fields, defines the links between tables. This enables you to automate the generation of joins at AQL level.

AQL links are expressed as:

```
Link[.Link[.Field]]
```

By simplifying the joins in such a manner, AQL also simplifies the creation of the majority of queries used for the AssetCenter database.

Example

The following query, written in AQL, returns for each model:

- Its **ID** (**lModelId**).
- Its **Full name** (**FullName**).
- The **Name** of the table linked to the brands (**amBrand**).

```
SELECT lModelId, FullName, Brand.Name FROM amModel
```

Here is the same query written in SQL Oracle or Microsoft SQL Server:

```
SELECT M1.lModelId, M1.FullName, B2.Name FROM amModel M1, amBrand B2 WHER
E M1.lBrandId=B2.lBrandId
```

The two joins between the **Models** table (**amModel**) and the Brands table (**amBrand**) are handled automatically in AQL. Using the AssetCenter's graphic query editor, you just have to click the fields of a selected table or linked table in a tree-structured list in order to create the corresponding AQL code.

Note:

On all systems except Oracle and DB2, the number of outer joins is limited to 1.

Under Microsoft SQL Server 7 and MSSQL 2000, you can modify the **amdb.ini** file to get round possible query-execution problems. Use the following instruction to modify this file in the detail of your connection:

```
=1
```

Reason for and usefulness of primary key 0 records

Primary key "0" records

The AssetCenter data module has certain specificities:

- The primary and foreign keys of each table are "32-bit integer" type fields.
- A foreign key that does not point to a record is set to "0" (and not "NULL").
- Each table has an empty record whose primary key is set to "0".

Usefulness

With these primary key "0" records, the results of a query using a non outer join between two given tables, A and B, can include records from table A which are not linked to any real record in table B (link not populated). These are records in table A, which are linked to the primary key "0" record in table B.

Example:

The following query, written in AQL, returns for each portfolio item's asset tag, the name of its user and its supervisor:

```
SELECT AssetTag, User.Name, Supervisor.Name FROM amPortfolio
```

A portfolio item that is not assigned to a user and/or supervisor appears in the results of the query. At database level, such an asset is linked to the primary key "0" record in the Departments and employees table.

Reason for these specificities

This section explains why primary key "0" records are used, whereas a query using an external SQL join can select records in table A that are not linked to a record in table B.

Primary key "0" records enable you to compensate for the fact that certain RDBMs do not handle multiple outer-joins: Using primary key "0" records, SQL queries generated from AQL queries do not use outer joins.

Example:

The AQL query below searches for each portfolio item, its asset tag and name of location of its user. The results will include assets that do not have a user and assets whose users do not have a location.

```
SELECT AssetTag, user.location.name FROM amPortfolio
```

If the generated SQL code used the outer joins of the DBMS, the SQL code generated for Sybase SQL Server would be:

```
SELECT a.AssetTag, l.name FROM amPortfolio a, amEmplDept e, amLocation l
WHERE a.lUserId *= e.lEmplDeptId AND e.lLocaId *= l.lLocaId
```

This code is not supported by Sybase SQL Server since it uses several outer joins one after another.

However, since there is a primary key "0" record in the table of departments and employees and locations, it is not necessary to use outer joins. AssetCenter thus generates SQL code without outer joins:

```
SELECT l.name FROM amPortfolio a, amEmplDept e, amLocation l WHERE a.lUse
rId = e.lEmplDeptId AND e.lLocaId = l.lLocaId
```

This query gives the expected results, since the **User** and **Location** links still point to a record in the table of departments and employees or locations (they point to the primary key "0" record if the link is not populated).

Consequences

- It is important to take these records into account in the queries that you write, especially when using aggregate functions.

Example:

```
SELECT count(AssetTag) FROM amPortfolio
```

If you execute the above query, which counts the number of assets in the table of assets, the primary key "0" record is taken into account in the results. You therefore need to decrease the results by 1 in order to obtain the real number of assets in the database.

- It is rarely necessary to have to generate outer joins at DBMS level.

Note:

Note: If you really want to generate outer joins at the DBMS level, use the "=*" and "*=" AQL operators.

Usage of NULL

AssetCenter uses the NULL value of the DBMS in two cases only:

- For an empty "Text" type field.
- For a non populated "Date" or "Date and time" type field.

AQL allows you to use several equivalent syntaxes, as shown below. It converts them to the equivalent valid SQL code of the database engine.

For empty "Text" type fields, you can use any of the following syntaxes, since the NULL value will be stored in the database:

WHERE <text field> = NULL

WHERE <text field> IS NULL

WHERE <text field > = ''

For non-populated "Date" or "Date and time" type fields, you can use any of the following syntaxes, since the NULL value will be stored in the database:

WHERE <field date or date+hour> = NULL

WHERE <field date or date+hour> IS NULL

WHERE <field date or date+hour> = []

Note:

Note: When a numeric field is not populated, its value is "0". Similarly, the absence of a link is described as "Link = 0" or "foreign key = 0". Example: "Location=0" or "lLocaId=0".

Self

"Self" is an expression that is equivalent to the description string on the table to which it is applied.

Using "Self" enables you to simplify queries and take any customization of the AssetCenter database into account.

Example:

If the description string of the table of departments and employees is:

```
[Name], [FirstName], ([Phone])
```

The AQL query:

```
SELECT self FROM amEmplDept
```

Is equivalent to:

```
SELECT (((((Name + ',') + FirstName) + '(') + Phone) + ')') FROM amEmplDept
```

CurrentUser

"CurrentUser" enables you to write queries that depend on the person connected to the database.

"CurrentUser" can be used as an expression, for example in a query, or as a link. You have to enter this expression manually as it is not offered by the query editor.

Used as "expression"

Example: We are looking for all the portfolio items used by the employee connected to the database.

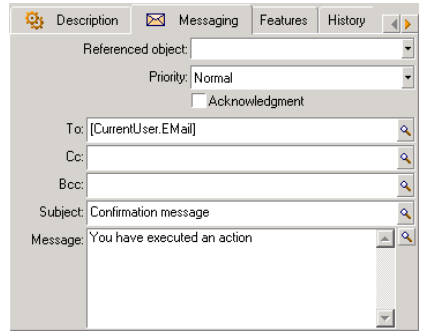
```
SELECT lPortfolioItemId FROM amPortfolio WHERE User = CurrentUser
```

Used as "link"

"CurrentUser" can be considered as a link that is found in every table and that points to the record corresponding to the current user in the table of departments and employees.

- With the form "CurrentUser", this function points to the record corresponding to the current user.
- With the form "CurrentUser.Field", this function returns the value of the field for the current user.

Example: When an action is triggered by the connected user, it is possible to contextually trigger another messaging type action, which automatically sends a warning message to the connected user. You just need to populate the detail of the action as follows:



System itemized-lists

If an AQL query uses a system itemized list, you must use the values that are stored in the database and not those which are displayed on screen.

Example:

The following query selects those contracts whose **Type** field (SQL name: seType) is set to **Master lease**:

```
SELECT Self FROM amContract WHERE seType = 1
```

The **Type** field (SQL name: seType) is a system itemized list. The values stored in the database are:

- 0 for **Other**
- 1 for **Master lease**
- 2 or **Lease schedule**
- 3 for **Insurance**
- 4 for **Maintenance**

Note:

The values of system itemized list can be found via AssetCenter Database Administrator, or the **database.txt** file, which describes the structure of the database.

This file is located in the following folder: **[AssetCenter installation folder]/doc/infos**

Hierarchic tables

All the hierarchic tables contain:

- A "FullName" field.
- A "sLvl" field.

"FullName" fields

For each record in a hierarchic table, the "FullName" field stores the value of a field of the record, preceded by a tree structure constituted by the values of the fields of parent records until root level.

Values are separated by the "/" character without spaces. This character also appears at the start and at the end of the tree-structure.

Examples:

- For the table of assets, the "FullName" field stores the Asset Tag of the asset preceded by the Asset Tag of its parent asset, that in turn preceded by the Asset Tag of its parent asset, and so on.

```
FullName = '/PC118/DD054/CR012/'
```

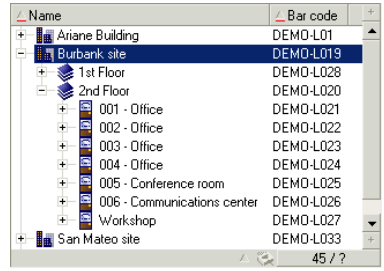
- In the table of locations, the "FullName" field stores the name of the location preceded by the names of parent locations.

```
FullName = '/Milwaukee/Water St. Site/Building A/5th floor/'
```

"sLvl" fields

For each record in a hierarchic table, the "sLvl" indicates its level in the tree-structure.

Root level is level 0.



The following query selects the "Sales Head Office" record and its sub-components:

```
SELECT Self FROM amEmplDept WHERE (FullName LIKE '/Sales Head Office/Sales/%') AND (sLvl >= 1)
```

The following query selects the "Sales Head Office" record but not its sub-components:

```
SELECT Self FROM amEmplDept WHERE (FullName LIKE '/Sales Head Office/Sales/%') AND (sLvl = 1)
```

The following query selects the sub-components of the "Sales Head Office" record but not the "Sales Head Office" record itself:

```
SELECT Self FROM amEmplDept WHERE (FullName LIKE '/Sales Head Office/Sales/%') AND (sLvl > 1)
```

Simplified AQL notation

This section lists the notation that can be used to simplify the writing of AQL queries:

Foreign keys

In clauses other than the SELECT and ORDER BY clauses, the SQL name of a link that is not followed by a period is equivalent to the SQL name of the associated foreign key.

Example: the clause:

```
WHERE location = 0
```

Is equivalent to:

```
WHERE lLocaId = 0
```

Where "location" is the SQL name of the "Location" link from the table of departments and employees to the table of locations, and "lLocaId" is the SQL name of the associated foreign key in the table of assets.

Description strings

In SELECT and ORDER By clauses, the SQL name of a link that is not followed by a period is equivalent to the join <SQL name of link >.self, which is in turn equivalent to <SQL name of link>.<Description string>.

Example:

If the description string of the table of departments and employees is:

```
[Name], [FirstName] ([Phone])
```

Then the AQL query:

```
SELECT user FROM amPortfolio
```

Is equivalent to the query:

```
SELECT user.self FROM amPortfolio
```

That is itself equivalent to:

```
SELECT (((((User.Name + ',') + User.FirstName) + '(') + User.Phone) + ')'
) FROM amPortfolio
```

Features

AQL gives you direct access to the features of a table, as if they were direct fields in the table. To search feature values for a given table, you just need to write the SQL name of the feature ("fv_" prefix).

Example: The following query searches the values of the feature with SQL name **fv_WorkUnit** for the **Departments and Employees** table (**amEmplDept**):

```
SELECT fv_WorkUnit FROM amEmplDept
```

Calculated fields

AQL facilitates the use of calculated fields associated with a table.

You just need to write the SQL name of the calculated field ("cf_" prefix).

Sorts and indexes

AQL allows two strategies for queries that use sorts (ORDER BY clause):

- A mode whereby AssetCenter forces the indexes indicated in the query, when used, and displays the results as and when they are retrieved.
- A mode whereby AssetCenter does not force the indexes indicated in the query. In this case, the DBMS determines how the data is sorted.

Note:

The choice between these two different methods is not available under SQL Anywhere. The database engine automatically selects the most suitable method.

Example

In the case of the following query:

```
SELECT lModelId, Brand FROM amModel ORDER BY Brand
```

- Access without **Forcing the indexes**: the database engine scans the table in full without using the "Brand" index indicated in the query. It searches all data items satisfying the query, sorts them according to the "Brand" and sends them to the user. The results will only be displayed after a certain period of time.

- In the other case: The database engine uses the "Brand" index, and displays the results as and when they are found. The first data items are thus shown rapidly, but the overall processing time is longer.

How to force the indexes

The way in which you force the utilization of the indexes depends on the way in which you create the query.

Via the Configure list menu

You can configure the data-access type for each list in AssetCenter. This is the case for both main lists and list contained in tabs. To do this:

1. Go to the list you want to configure.
2. Right-click.
3. Select **Configure list** from the shortcut menu.
4. In the **Columns and sort** tab, check the **Force indexes** box in order to use the indexes indicated in the query and display the results as and when they are generated. Uncheck this box in order to select the other type of access.

Using AQL

If you write a query directly in AQL, you can force the use of indexes via the "FIRST_ROWS" clause.

Example:

```
SELECT FIRST_ROWS AssetTag FROM amAsset ORDER BY AssetTag
```

Note:

If the sort focuses on the system itemized lists (for example, on the Features table in the **seDataType** field), there is a possibility that the sort is not valid when an index is forced.

Sort order

The sort order depends on:

- The database engine.
- The use of indexes.

Under Oracle for WorkGroups

With indexes forced

- NULL records do not appear.
- The sort is performed according to the value of the ASCII codes thus differentiating between upper and lower case characters (binary sort).

Without indexes forced

- NULL records are displayed.
- Oracle for WorkGroups is not case sensitive.

Example

Sort

Starting list	A B C D a b NULL NULL
List with indexes forced	A B C D a b
List without indexes forced	NULL NULL A a B b C D

Under Microsoft SQL Server or Sybase SQL Server

The sort order depends on a parameter set when the database is created. These engines can be configured in order to be case sensitive or to take accented characters into account, etc.

Under Sybase SQL Anywhere

Under Sybase SQL Anywhere, the indexes cannot be forced via an AQL query.

The database engine determines the optimal method used to access the data and to sort it.

Precautions

With complex queries it often difficult to determine whether it is more "advantageous" to force the indexes or not. In practice, we recommend performing tests before making a final decision.

In particular, we recommend testing with and without the indexes forced in the case of a list that is filtered, be it explicitly (via a simple filter, query) or implicitly (via access restrictions).

Query editor

AssetCenter includes a query editor. This tool enables you to design queries and preview their results. It is particularly aimed at database administrators and power users.

Overview

The query editor makes it possible to design queries:

- Either by using the graphic interface (assisted query design).
- Or by writing the query directly in AQL.

Whether you use the graphic method or prefer to write directly in AQL (both approaches are frequently combined), you get to see a real-time transcription of your query in SQL. However, you cannot write your queries directly in SQL.

Figure 3.1. Query editor - composition modes

Using the query editor, a power user or an administrator can create, modify or delete AQL queries. These queries can be used in the appropriate context by their author or other users.

Accessing the query editor

You can access the query editor in different ways:

- Via the **Tools/ Queries** menu item. Using this menu, you can create queries for your own use, which can also be used freely by other users. The queries can be executed:
  - Either directly via the window displayed by the **Tools/ Queries** menu item.

  - Or by when using a "query filter" when displaying the main table of the query.
- Via the numerous functions of AssetCenter that call on queries: Access restrictions, query filters, list configuration, etc.
- Via external programs: AssetCenter Export, etc.

The version of the query editor that is shown is simplified to a certain degree according to the context.

Example: Let's suppose that you have a query such as the following:

```
SELECT [FIRST_ROWS] <field>[, <field>...] FROM <table> [WHERE <clause>] [
ORDER BY <clause>]
```

In the simplified versions of the query editor (simple filters, query filters, etc.), you only have to define the WHERE clause of the query. The other components of the query (starting table, fields, etc.) are implicit. For example, in the case of a query filter, the table is that on which the filter is applied, the fields and the sort conditions are the columns and the sort conditions which are defined via the **Configure list** shortcut menu item. The same is true for the query editor accessed via the **Tools/ Queries** menu item.

Thus, the following query given in full:

```
SELECT self FROM amModel WHERE Brand.Name='Compaq'
```

is written as follows when used in a query filter (only the WHERE clause is explicitly given) used on the table of models:

```
Brand.Name='Compaq'
```

On the other hand, the **Configure list** command enables you to access a more comprehensive version of the query editor:

- The **Columns and sort** tab defines the fields to be displayed in columns and the sort conditions (these sort conditions correspond to the ORDER BY clause).
- The **Force indexes** box replaces the FIRST_ROWS clause in the SQL code.
- The **Filter (WHERE clause)** tab defines the "WHERE" clause.
- The table is implicit.

Creating a query using the query editor

To create a query using the query editor, select the **Tools/ Queries** menu item. The window that is displayed has two tabs, **Filter (WHERE clause)** and **Preview**:

- The **Filter (WHERE clause)** tab is a graphical interface that determines the conditions of your query. It defines the elements of the WHERE clause.
- The **Preview** tab displays the transcription of your query in SQL code and enables you to test it.

Step 1: Populate the fields at the top of the query detail.

You must specify the starting table of your query.

If you want the query you are creating to be able to be accessible to other users, uncheck the **Not shared** option (SQL name: bPrivate).

Note:

The administrator has access to all queries stored in the database, even those queries that are **Not shared**.

Once you have filled in the basic information on the query, click **Create** to be able to access the detail tabs of the query.

Step 2: Define the filter conditions in the Filter (WHERE clause) tab.

The AssetCenter query editor enables you to define conditions that combine fields, calculation expressions, constants and operators.

You can define one or more filter conditions.

To define a filter condition:

1 From the starting table, select a field, constant or expression (**Field 1**), that you will compare with a field, constant or expression (**Field 2**).

2 Confirm the filter condition by transferring it to the lower part of the screen using the  button.

3 Confirm the query by clicking **Modify** in the query detail.

To define several filter conditions linked by the AND and OR logical operators:

1 Create a first filter condition as indicated above.

2 Define the other filter conditions and confirm them using the **AND** or **OR** buttons.

3 Confirm the query by clicking **Modify** in the query detail.

Note:

In order to make a modification to the selected conditions, click the  button to delete the contents of the window, or else modify the AQL code directly.

Note:

In place of the graphic tool, you can enter your query directly in AQL in the zone at the bottom of the **Filter (WHERE clause)** tab.

Step 3: Preview the execution of the query

To test the query and see its transcription in SQL language:

1 Go to the **Preview** tab in the query detail.

2 Click the  icon: AssetCenter shows a preview of the results of the query, in the form of a list of records. The number of records satisfying the query is shown at the bottom right of the window.

Note:

The SQL code contained in the **Preview** tab cannot be modified directly.

Fields used in queries

When defining filter conditions in queries, you can call on:

- A field in the table concerned by the query.
- A linked field.
- Features associated with the table.

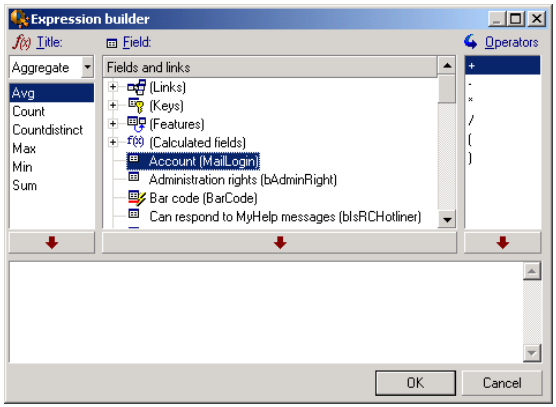
Writing expressions

Expressions enable you to perform calculation operations in your query. For example, you can use the "Count" function to count the number of resultant records of a query.

To write an expression, you can either:

- Enter it directly in the corresponding field.
- Or use the AssetCenter expression builder.

To use the expression builder, click the button adjacent to the edit zone of the **Filter (WHERE clause)** tab of the query detail.



The expression builder comprises three columns:

- The "Function" column lists existing AQL functions. Clicking applies a filter on the list of AQL functions according to their type: "Aggregate", "String", "Date", "Numeric", "Test".
- The "Field" column lists the different fields that can be used in a query.
- The "Operators" column lists the operators that can be used in the expression.

To insert a "Function", "Field" or "Operator" in the expression:

1 Select the function, field or operator.

2 Click .

Once the expression is defined, click **OK** to transfer it over to the **Filter (WHERE clause)** tab in the query detail.

Constants

Constants are fixed values that you assign to selection criteria. For example, if you are searching all models with brand **3Com**, you assign the constant value **3Com** to the **Brand.Name** linked field in the table of models.

To select constant:

1 Click the icon.

2 A selection window is displayed that shows the values present in the database for the field specified as search condition.

Note:

Even in the case of "Itemized list" type fields, the window displayed by clicking the icon only shows those values used in the database.

AQL syntax

Using AQL requires familiarity with SQL language. However, a detailed description of the syntax of SQL is beyond the scope of this manual. For further information, please consult the appropriate reference documentation.

Conventions

Here are the conventions used to describe the syntax of AQL:

Table 3.1. AQL - syntax conventions

[ ]	These brackets surround optional items. Do not type in these brackets.
< >	These brackets surround logical items. Do not type in these brackets.
|	The vertical bar indicates that choices are mutually exclusive.
. . .	The ellipsis indicates that the preceding text may be repeated once or several times.
FROM	Terms in uppercase letters indicate literal expressions.

Syntax of queries

Simple queries

SELECT [DISTINCT] [FIRST_ROWS] <liste de sélection>
[FROM clause [page 50]]
[WHERE clause [page 51]]
[GROUP BY clause [page 52]]
[HAVING clause [page 53]]
[ORDER BY clause [page 54]]

Sub queries

AQL supports the use of sub-queries in the place of fields.

Note:

In sub-queries, the SELECT statement only authorizes a single expression.

(**SELECT** [DISTINCT] <expression>
[FROM clause [page 50]]
[WHERE clause [page 51]]
[GROUP BY clause [page 52]]
[HAVING clause [page 53]]
)

Warning:

Sub-queries must be contained with parentheses.

Example of utilization:

```
SELECT Self FROM amAsset WHERE mPrice >= (SELECT Max(mPrice)/2 FROM amAss
et)
```

UNION type queries

UNION enables you to group together the results of several queries:
SELECT <selection list>
[FROM clause [page 50]]

[WHERE clause [page 51]]
[GROUP BY clause [page 52]]
[HAVING clause [page 53]]
[**UNION | UNION ALL | INTERSECTS | MINUS**]
SELECT <selection list>
[FROM clause [page 50]]
[WHERE clause [page 51]]
[WHERE clause [page 51]]
[GROUP BY clause [page 52]]
[HAVING clause [page 53]]...]
[ORDER BY clause [page 54]]

Elements of a query

Fields and links

Queries involve fields and links in the AssetCenter database.

You can indicate the name of a field:

- In reference to the starting table of the query. In this case, it is not necessary to specify the name of this table:

[Link. ...[Link.]]<field>

Examples from the **Portfolio items** table (**AmPortfolio**):

```
Model
User.Name
User.Location.Name
```

- As an absolute reference. In this case, you need to indicate the name of the table to which the field belongs:
  - Either by declaring the table in the **FROM** clause and using its name (or a possible alias):

    <table.[link...]field>

    <alias.[link...]field>
  - Or by not declaring the table in the **FROM** clause and instead using ":":

    <table.[link...]field>

    <table[_alias]:[link[_alias]...]field>

These last two notations are particularly useful if you cannot use the **FROM** clause.

For example, when writing a query in AssetCenter, you only have access to the **WHERE** clause. The starting table of the query is implicit (table on which a filter is applied, **Table** field (**TableName**) in query detail, etc.). However, you may need to use other tables in the query. In this case, the ":" notation allows you accomplish this.

Constants

The following syntax is valid for the constants that may be involved in queries.

Numeric constants

The period is used as decimal separator.

Examples:

12

52.23

Text type constants

They are contained within single quotes.

Examples:

'Computer'

'Monitor'

Date or time type constants

Date or time type constants are contained between # characters. Their format must respect the following rules:

- Years are expressed on 4 figures.
- Dates are expressed as Year-Month-Day.
- Time is expressed as Hours-Minutes-Seconds.
- The 24 hour clock is used (and not the 12 hour clock with A.M. or P.M.).
- The separator used in dates is the "/" or "-" character.
- The separator used for time is the ":" character.
- Months, days, hours, minutes, and seconds; these are expressed as 2 figures.
- When date and time are expressed together, the data always precedes the time and they are separated by a space.

Examples:

#yyyy-mm-dd hh:mm:ss#

#yyyy-mm-dd#

#hh:mm:ss#

#2004-01-01 01:00:03#

Expressions

Expressions are formed using:

- Constants
- Fields
- Functions
- Sub-queries

You can combine these elements with operators and parentheses in order to build complex expressions.

Comparison expressions take the form:

<expression> <comparison operator> <expression>

Logical expressions take the form:

<comparison operator> <AND | OR> <comparison operator>

You can use parentheses to combine several logical expressions.

Operators

Logical operators

Logical operators are applied to link two expressions:

Table 3.2. AQL - logical operators

Operator	Meaning
AND	Logical "AND"
OR	Logical "OR"

In order to optimize a query, it is sometime wise to avoid logical operators if a comparison operator can be used instead. The following example illustrates how to optimize a query filter used to select portfolio items whose **Assignment**

field (SQL name: seAssignment) is set to **Awaiting delivery** or **Return for maintenance**. The values of these two elements of a system itemized list are "3" and "4" respectively. It is therefore possible to write:

```
(seAssignment=3) OR (seAssignment =4)
```

The last value of the system itemized list being "4", it is preferable to write the query as follows:

```
seAssignment >=3
```

Comparison operators

Comparison operators are used to compare two expressions.

Table 3.3. AQL - comparison operators

Operator	Meaning
=	Equals
<>	Different than
=!	
>	Greater than
<	Less than
>=	Greater or equal to
=<	Less than or equal to
=*	Right outer join. Because of the way in which AQL handles links, the use of this operator is limited
*=	Left outer join. Because of the way in which AQL handles links, the use of this operator is limited.

Operator	Meaning
LIKE	Works like the = operator and allows you also to use "wildcard" characters.
NOT LIKE	The following "wildcard" characters are available:
	"%" replaces any character string.
	"_" replaces any single character.
	Depending on the database engine used (SQL Anywhere, SQL Server and Sybase support it, Oracle for WorkGroups doesn't):
	[abc?] lets you define a list of possible characters (no space between the possible values.)
	[a-c] lets you define a range of possible values.
	DB2 does not support use of the LIKE X operator, if X includes a SQL column name. Only constants are supported for this operator. For example, the following query is not correct for DB2:
	SELECT COL1, COL2 **FROM** TABLE1 **WHERE** COL1 LIKE COL2
IS NULL	Tests whether the value of a field is "NULL" or not.
IS NOT NULL	AssetCenter only authorizes the "NULL" value for empty text fields and for **Date** or **Date+Time** fields that are not populated.

Note:

SQL Anywhere is not able to process "LIKE X" clauses when X contains more than 128 characters. If X is larger than 128 characters applying the query provokes an ODBC error message. This problem can, for example, occur when displaying lists in tree view since this operation uses a "FullName" field.

Operators specific to sub-queries

You can compare a value to the results of a sub-query using the following operators:

- **= ANY (sub-query)**
- **= ALL (sub-query)**
- **= SOME (sub-query)**

Example:

- The following query provides the list of portfolio items whose brand is used at the Milwaukee site:

```
SELECT lModelId, Model.Brand FROM amPortfolio WHERE Model.Brand = ANY (SE
LECT Model.Brand FROM amPortfolio WHERE Location.FullName = '/Milwaukee s
ite')
```

Selection list

Selection lists define the items to be extracted or displayed. They specify the SELECT statements in queries.

A selection list is made up of one or more expressions separated by commas:

<expression> [,<expression>...]

Each expression can be linked to an alias. For example:

```
SELECT MrMrs, (Name + FirstName) Identity FROM amEmplDept
```

This is particularly useful at the level of export queries to attribute a name to the exported columns.

Note:

Certain DBMSs limit the number of expressions in a given SELECT statement.

FROM clause

The **FROM** clause indicates the table or tables concerned by a **SELECT** statement.

AQL authorizes the utilization of aliases for table names.

Syntax

FROM <table name> [table alias] [, <table name> [table alias>] ...]

Starting table of a query

The first table indicated in the **FROM** clause of a query is the starting table of the query.

If a query uses a field whose table is not specified, AQL considers that it belongs to the starting table of the query. The AQL **FROM** clause differs from the clause with the same name in SQL.

For example, in the following sub-query, AQL searches the **AssetTag** field in the **amAsset** table:

```
SELECT AssetTag FROM amAsset
```

Number of tables in a query

The number of tables authorized in a query depends on the DBMS used.

Example:

- Oracle: You can use as many tables as you like.
- Microsoft SQL Server or Sybase SQL Server: You are limited to 16 tables in a query.

Warning:

When counting the number of tables in a query, do not forget to take into account those tables that are not explicitly mentioned, in particular if the query uses links. Also look out for the "fv_" notation (search of feature values) which generates an additional join at DBMS level. Similarly, the "cf_" notion (calculated fields) can generate additional joins.

Examples

```
FROM amPortfolio
FROM amPortfolio a, amLocation l
```

The following queries are equivalent:

```
SELECT AssetTag FROM amAsset
SELECT a.AssetTag FROM amAsset a
SELECT amAsset.AssetTag FROM AmAsset
```

WHERE clause

The AQL **WHERE** clause is equivalent to the **WHERE** clause in SQL.

It specifies the search conditions, specifying the elements to extract from the database. These conditions can also express themselves in **WHERE** or **HAVING** clauses.

Syntax

WHERE <search conditions>

Writing search conditions

In the majority of cases, you will need to write the conditions using the following form:

```
<WHERE | HAVING> [NOT] <expression> <comparison operator> <expression>
<WHERE | HAVING> [NOT] <logical expression>
<WHERE | HAVING> [NOT] <field> [NOT] LIKE 'xxxxx'
<WHERE | HAVING> [NOT] <logical expression> <AND | OR> <logical expressio
n>
<WHERE | HAVING> [NOT] <field> IS [NOT] NULL
```

In some other case, you may need to write more complex queries, such as:

```
<WHERE | HAVING> [NOT] EXISTS (<sub-query>)
<WHERE | HAVING> [NOT] <expression> [NOT] IN (<list of values> | <sub-que
ry>)
<WHERE | HAVING> [NOT] <expression> <comparison operator> <ANY | ALL> (<s
ub-query>)
```

GROUP BY clause

The AQL **GROUP BY** clause is equivalent to the **GROUP BY** clause in SQL.

Syntax

GROUP BY <expression without aggregates> [, <expression without aggregates>]...

Writing tips

GROUP BY specifies subsets of the table. The subsets are defined in the **GROUP BY** clause by an expression, which can be the name of a field, for example.

If aggregate functions are included in the selection list of the **SELECT** statement, **GROUP BY** searches the resulting value for each subset. These resultant values can be used in a **HAVING** clause.

When a query makes use of the **GROUP BY** clause, each expression of the selection list must provide a single value for each subset.

GROUP BY - Examples

The following query gives the total number of brands present in the database. For each asset with an associated brand, AssetCenter returns an instance on the brand.

```
SELECT Count(Model.Brand.Name) FROM amAsset
```

By using the **GROUP BY** clause, we obtain a list of brands and the number of assets of each brand:

```
SELECT Model.Brand.Name, count(lAstId) FROM amAsset GROUP BY Model.Brand
```

HAVING clause

The AQL **HAVING** clause is equivalent to the SQL **HAVING** clause.

Syntax

HAVING <Search conditions>

Differences with the WHERE clause

It specifies the search conditions like the **WHERE** clause. However, these two clauses differ as follows:

- The **HAVING** clause specifies the restrictions to be applied to aggregate functions in the selection list. The restrictions affect the number of resultant lines but do not affect the calculations linked to aggregate functions.
- When the query uses an **WHERE** clause, the search conditions restrict the lines subject to aggregate calculation functions but do not affect the resultant lines.

Examples

Example of query where the **WHERE** clause is equivalent to the **HAVING** clause:

The following query returns the list of brands whose name starts with a letter after the letter **B** and the number of asset of each of these brands:

```
SELECT Model.Brand.Name, count(lAstId) FROM amAsset GROUP BY Model.Brand.
Name HAVING Model.Brand.Name > 'B'
```

It is also possible to express the same query using the **WHERE** clause:

```
SELECT Model.Brand.Name, count(lAstId) FROM amAsset WHERE Model.Brand.Nam
e > 'B' GROUP BY Model.Brand.Name
```

Example of query using the **HAVING** clause:

The **HAVING** clause enables you to use aggregate functions (such as **Count**); This is not the case with the **WHERE** clause. Thus, the following query searches all brands represented by more than one asset:

```
SELECT Model.Brand.Name, count(lAstId) FROM amAsset GROUP BY Model.Brand.
Name HAVING count(Model.Brand) > 1
```

ORDER BY clause

The AQL **ORDER BY** clause is equivalent to the SQL **ORDER BY** clause.

Items can be sorted:

- In ascending order: **ASC**. This is the default sort order.
- In descending order: **DESC**.

Syntax

ORDER BY <expression> [**ASC** | **DESC**] [, <expression> [**ASC** | **DESC**]...]

INSERT clause

This clause enables you to insert one or more records into a database table.

Syntax

INSERT INTO <Table name> [table alias] (<Name of a field> [, <Name of a field>]...) **VALUES** (<expression> [, expression]...) | **AQL sub-query**)

This clause is included in the AssetCenter API **AmDbExecAql**.

For more information about the AssetCenter API, refer to the Programmer's reference guide, chapter **Alphabetic reference**.

Example

The **INSERT** clause can simplify the code of a **Supplemental delivery information** wizard:

Wizard code not using the INSERT clause

```
hrAlarm = AmCreateRecord("amDateAlarm")
            lErr = AmSetFieldLongValue(hrAlarm, "bSecondLevel", 0)
            lErr = AmSetFieldLongValue(hrAlarm, "dtTrig1", AmGetFieldLong
```

```
Value(hrAsset, 2)-lDaysBefore*86400)
            lErr = AmSetFieldLongValue(hrAlarm, "lAction1Id", lActionId)
            lErr = AmSetFieldLongValue(hrAlarm, "lMonitObjId", lAstId)
            lErr = AmSetFieldStrValue(hrAlarm, "MonitoredField", "dWarrEn
d")
            lErr = AmSetFieldStrValue(hrAlarm, "MonitoredTable", "amAsset
")
            lErr = AmSetFieldLongValue(hrAlarm, "sDaysBefore1", lDaysBefo
re)
            lErr = AmInsertRecord(hrAlarm)
```

Wizard code using the INSERT clause

```
lErr = AmDbExecAql("insert into amDateAlarm (bSecondLevel, dtTrig1, lActi
onId, lMonitObjId, MonitoredField, MonitoredTable, sDaysBefore1) values (
 0, " & AmGetFieldLongValue(AmGetFieldLongValue(hrAsset, 2)-lDaysBefore*8
6400 & ", " & lAstId & ", 'dWarrEnd', 'amAsset', " & lDaysBefore & ")")
```

UPDATE clause

This clause enables you to update one or more fields of a record in a database table.

Syntax

UPDATE <table name> [table alias] **SET** (<name of a field> [, <name of a field>...]) [FROM clause [page 50]] [WHERE clause [page 51]]

Example

The **UPDATE** clause can help simplify the code of an action that triggers an order action :

Action code not using the UPDATE clause

```
hr = AmGetRecordFromMainId("amPOrder", [lPOrdId])
      lErr = AmSetFieldLongValue(hr, "seStatus", "$(IDS_POSTATUS_ORDERED)
")
      lErr = AmUpdateRecord(hr)
```

Action code using the UPDATE clause

```
lErr = AmDbExecAql("update amPOrder set seStatus = 21 where lPOrdId = " &
[lPOrdId])
```

DUPLICATE clause

This clause enables you to duplicate a record existing in a database table.

This function is particular to AssetCenter.

For more information, refer to the **User interface** guide, chapter **Operations on records**, secstion **Duplicating a record**.

Syntax

DUPLICATE <table name> [table alias] **SET** (<name of a field> [, <name of a field>...]) [FROM clause [page 50]] [WHERE clause [page 51]]

DELETE clause

This clause enables you to delete the fields of a record in a database table.

Syntax

DELETE [FROM clause [page 50]] [WHERE clause [page 51]]

AQL function reference

The following AQL functions can be used in queries and formulas:

- Aggregate type AQL functions
- String type AQL functions
- Date type AQL functions
- Numeric type AQL functions
- Test type AQL functions

Note:

You can also use native SQL functions of your DBMS. In this case, the resulting code is not portable.

Aggregate-type AQL functions

Table 3.4. AQL - Aggregate-type functions

Function	Description
Avg(<column>)	Returns the average value of a "number" type column. Returns "0" if the column does not have any records.
Count(<Column>)	Counts the non-null values in a column.
Countdistinct(<Column>)	Counts the distinct non-null values in a column.
Max(<column>)	Returns the maximum value of a "Number", "Text" or "Date" type column. If the column does not have any records, returns "0" ("number" type column), "empty string" ("Text" type column), or "empty date" ("Date" type column).
Min(<Column>)	Returns the minimum value in a "Number", "Text" or "Date" type column. If the column does not have any records, returns "0" ("Number" type column), "empty string" ("Text" type column), or "empty date" ("Date" type column).
Sum(<Colonne>)	Returns the sum of the values of a "number" type column. Returns "0" if the column does not have any records.

These functions jointly use the "GROUP BY" and "HAVING" clauses.

String-type AQL functions

Table 3.5. AQL - String-type functions

Function	Description
Ascii(<String>)	Returns the ASCII value of the first character of the <string>.
Char(<n>)	Returns the character with ASCII code "n".
Left(<String>, <n>)	Returns the "n" first characters of the <string>.
Lower(<String>)	Returns the <string> in lowercase.
Ltrim(<String>)	Removes the spaces at the left of the <string>.
Right(<String>, <n>)	Returns the "n" last characters of the <string>
Rtrim(<String>)	Removes the spaces at the right of the <string>

Function	Description
Substring(<String>, <n1>, <n2>)	Extracts the sub-string starting at character "n1" in the <string> and with length "n2" (the 1st character of the <string> is considered as character number 1).
Upper(<String>)	Returns the <string> in uppercase.

Date-type AQL functions

Table 3.6. AQL - Date-type functions

Function	Description
Year(<date>)	Returns the number representing the year for a "Date" or "Date and time" type field (e.g: 2000).
Month(<date>)	Returns the number of the month for a "Date" or "Date and time" type field (1=January, ..., 12=December).
Day(<date>)	Returns the number of the day in the month for a "Date" or "Date and time" type field (1-31).
DayOfYear(<date>)	Returns the number of the day in the year for a "Date" or "Date and time" type field (1-366).
WeekDay(<date>)	Returns the number of the day in the week for a "Date" or "Date and time" type field. This number depends on how the server is configured. For example, the default configuration under Sybase or Microsoft SQL Server is (1=Sunday, 2=Monday, ..., 7=Saturday). The default configuration under Oracle is (1=Monday, ..., 7=Sunday).
Hour(<time>)	Returns the number of the hour in the day for a "Time" or "Date and time" type field (0-23).
Minute(<time>)	Returns the number of minutes for a "Time" or "Date and time" type field (0-59).
Second(<time>)	Returns the number of seconds for a "Time" or "Date and time" type field (0-59).
Getdate()	Returns the server's current system date.
AddDays(<date>, <number>)	Adds a given number of days to a "Date" or "Date and time" type field.
AddHours(<date>, <number>)	Adds a given number of hours to a "Date" or "Date and time" type field.
AddMinutes(<date>, <number>)	Adds a given number of minutes to a "Date" or "Date and time" type field.

Function	Description
AddSeconds(<date>, <number>)	Adds a given number of seconds to a "Date" or "Date and time" type field.
DaysDiff(<date1>, <date2>)	Number of days between the dates date1 and date2 ("floating point" number with decimals)
HoursDiff(<date1>, <date2>)	Number of hours between the dates date1 and date2 ("floating point" number with decimals)
MinutesDiff(<date1>, <date2>)	Number of minutes between the dates date1 and date2 ("floating point" number with decimals)
SecondsDiff(<date1>, <date2>)	Number of seconds between the dates date1 and date2 ("floating point" number with decimals)
DbToLocalDate(<date>)	Converts a date expressed in the time zone of the database server to a date expressed in the time zone defined at client machine level.
LocalToDbDate(<date>)	Converts a date expressed in the time zone of the client machine to a date expressed in the time zone of the database server.

Table 3.7. AQL - Examples of Date-type functions

Description	AssetCenter query language
All records modified during the last week	AddDays(dtLastModif,7)>=Getdate()
All work orders notified in the last hour	HoursDiff(Getdate(), dtNotif) <= 1 or AddHours(dtNotif, 1) >= Getdate()
All work orders closed in the last half-hour	MinutesDiff(Getdate(), dtActualFixed) <= 30 or AddMinutes(dtActualFixed, 30) >= Getdate()

The following query lists the work orders effectively carried out and resolved during the same day. The time zone of the client machine is taken into account:

```
SELECT Self FROM amWorkorder WHERE DayOfYear(DbToLocalDate(dtActualFixStart)) = DayOfYear(DbToLocalDate(dtActualFixed))
```

The following query lists all work orders that have effectively been started today:

```
SELECT Self FROM amWorkorder WHERE DayOfYear(DbToLocalDate(dtActualFixStart)) = DayOfYear(DbToLocalDate(GetDate()))
```

Numeric-type AQL functions

Table 3.8. AQL - Numeric-type functions

Function	Description
Abs(<Number>)	Returns the absolute value of a "number".
Ceil(<Number>)	Returns the smallest integer greater or equal to a "number".
Floor(<Number>)	Returns largest integer less than or equal to a "number".
Mod(<a>, <b>)	Returns the remainder of the division of "a" by "b" (a = qb + r, with q integer and 0 < r < q).
Round(<a>, <n>)	Rounds "a" to "n" decimal places.
Trunc(<a>, <n>)	Truncates "a" to "n" decimals.

Examples of application:

Abs (2.516) = 2.

Ceil (2.516) = 3.

Floor (2.516) = 2.

Mod (6,4) = 2.

Round (31.16, 1) = 31.20.

Round (31.16, 0) = 31.00.

Round (31.16, -1) = 30.00.

Trunc (31.16, 1) = 31.1.

Test type AQL functions

Table 3.9. AQL - Test-type functions

Function	Description
IsNull(<a>, <b>)	If "a" is "Null", replaces "a" by "b". The types of "a" and "b" must be compatible.

Examples of queries

Each of the following examples deal with a specific aspect of query design. You can modify or combine these examples to use them as the basis of your own queries.

These examples give the full syntax of the query. If you want to test them out as is, use AssetCenter Export. You will have to modify the syntax of these examples to use them in a query filter, for example.

Thus, the following query given in full:

```
SELECT self FROM amAsset WHERE Model.Brand.Name='Compaq'
```

is written as follows when used in a query filter (only the WHERE clause is explicitly given) used on the table of assets:

```
Model.Brand.Name='Compaq'
```

Further examples of queries are included in the demonstration database supplied with AssetCenter.

Note:

To view the transcription of a query in the corresponding SQL code of the DBMS you are using, display the **Preview** tab in the query detail.

To compare a field in the main table with a value

Example: All "Compaq" brand portfolio items.

```
SELECT Self FROM amPortfolio WHERE Model.Brand.Name = 'Compaq'
```

To compare a link in the main table with another link

Example: All portfolio items with the same location as their parent asset.

```
SELECT Self FROM amPortfolio WHERE Location = Parent.Location
```

To compare a link in the main table with a value

Example: All departments and employees directly linked to the "Burbank Agency".

```
SELECT Self FROM amEmplDept WHERE Parent.Name = 'Burbank Agency'
```

To compare according to a field in a table linked to the main table

Example: All portfolio items that have the same location name as their parent.

```
SELECT Self FROM amPortfolio WHERE Location.Name = Parent.Location.Name
```

Hierarchic tables

Utilization of "FullName" field

Example: All sub-locations of the location named "Ariane Building":

```
SELECT Self FROM amLocation WHERE FullName LIKE '/Ariane Building/%'
```

Utilization of "FullName" and "sLvl" fields

Queries on the hierarchic tables often make use of the "FullName" and "sLvl" fields.

Example: All the sub-locations of the location "Ariane Building", with a hierarchic level less than 3.

The hierarchic value of the root level of a tree-structure is equal to "0".

```
SELECT Self FROM amLocation WHERE (FullName LIKE '/Ariane Building/%') AN
D (sLvl < 3)
```

Pay special attention to the "/" characters that appear at the start and end of full names.

Query combining two conditions

Example: All employees with title "Account executive" and located at the "Burbank site".

```
SELECT Self FROM amEmplDEpt WHERE (Title = 'Commercial') AND (Location.Na
me = 'Madison site')
```

Comparison of a field with numbers, dates or text

Example: All work orders carries out between January 1, 2003 and December 31, 2003.

```
SELECT self FROM amWorkOrder WHERE (dtActualFixStart >= #2003-01-01 00:00
:00#) AND (dtActualFixStart <= #2003-12-31 00:00:00#)
```

Query concerning a feature

Example: All portfolio items whose feature with SQL name **fv_Size** showing a value greater than or equal to 150 cm.

```
SELECT Self FROM amPortfolio WHERE fv_Size >= 150.00
```

To search records according to an expression

Example: All assets whose purchase price is equal to the greatest purchase price contained in the database. Note that a sub-query is used in the main query in order to identify the maximum price.

```
SELECT Self FROM amAsset WHERE mPrice = (SELECT max(mPrice) FROM amAsset)
```

To search a field that is not populated

Example: All employees without a telephone number. Note than an empty string is represented by two single quotes ''.

```
SELECT Self FROM amEmplDept WHERE Phone=''
```

To search for the absence of a link

Case of a 1 link

Example: All portfolio items that have not been assigned to a user. Note that the absence of a link is denoted by "0".

```
SELECT Self FROM amPortfolio WHERE User = 0
```

Case of n links

Example: All models with no associated assets:

```
SELECT self FROM amModel WHERE NOT ( EXISTS (SELECT A1.lAstId FROM amAsse
t A1 WHERE A1.lModelId = amModel.lModelId))
```

This query scans the Models table, takes each model one after the other and compares the number of assets belonging to this model with 0.

Example combining a test on a 1 link and an n link

Example: All models without parent model or sub-model:

```
SELECT self FROM amModel WHERE (NOT ( EXISTS (SELECT A1.lModelId FROM amM
odel A1 WHERE A1.lParentId = lModelId))) AND (Parent = 0)
```

This query performs:

- A test on a 1 link ("Parent = 0"), to select those models without parent assets.
- A test in an n link ("0 = (SELECT COUNT(a.lModelId) FROM amModel a WHERE a.lParentId = lModelId)"), to select those models without sub-models. The test on the n link takes each model, selects its identifier "lModelId", and counts all the models whose "lParentId" identifier is equal to "lModelId".

Another example

All models without "Computer" natured sub-model:

```
SELECT self FROM amModel p WHERE NOT ( EXISTS (SELECT lModelId FROM amMod
el WHERE (FullName LIKE (p.FullName + '%/')) AND (Nature.Name = 'Computer
')))
```

Note:

If you try this query with AssetCenter Export, an error message will appear. You can ignore this message. The query actually works correctly.

Query with alias

Example: All employees having taken the 'Peregrine' training program and the a 'Database' training program.

Starting table: The table of departments and employees.

The query is as follows:

```
SELECT Self FROM amEmplDept WHERE (Trainings_1.Title = 'Peregrine') AND (
Trainings_2.Title = 'Base de données')
```

Aliases, expressed as "Training_1" and "Training_2" enable you to define 2 conditions concerning the 2 different records linked by the "Training" link.

If we had written:

```
SELECT Self FROM amEmplDept WHERE (Trainings.Title = 'Peregrine') AND (Tr
ainings.Title = 'Base de données')
```

We would have selected all employees having taken a training course with both titles.

If we had written:

```
SELECT Self FROM amEmplDept WHERE (Trainings.Title = 'Peregrine') OR (Tra
inings.Title = 'Base de données')
```

We would have selected all employees having taken one training course with one of the two titles.

4 Forms

CHAPTER

This chapter explains how to design forms with AssetCenter.

Use the **Tools/ Reporting/ Forms** menu item to display the list of forms.

Definition of a form

A form is a document model that lets you print data.

Unlike Seagate Crystal reports, forms are created directly in AssetCenter.

Installing preconfigured forms into your working database

AssetCenter is provided with preconfigured forms. These are already installed in the demonstration database, but you must import them into your working database, either when you create it or when you update it.

Importing forms during database creation

To import forms during the database creation:

1 Launch AssetCenter Database Administrator.
2 Select the **File/ Open** menu.
3 Select the **Open database description file - create new database** option.
4 Select the **gbbase.xml** file, located in the **config** sub-folder of the AssetCenter installation folder.
5 Select the **Action/ Create database** menu.
6 Populate all the fields that enable you to create a database. Consult the **Administration** guide, chapter **Creating a AssetCenter database** to find out what fields are necessary.
7 Select the option **Import extra data**.
8 In the **Data to import** frame, select **Forms**.
9 Click **Create** to validate the creation of the database.

Importing forms in an existing database

To import forms into an existing database:

1 Launch AssetCenter Database Administrator.
2 Select the **File/ Open** menu.
3 Select the **Open database description file - create new database** option.
4 Select the **gbbase.xml** file, located in the **config** sub-folder of the AssetCenter installation folder.
5 Select the **Action/ Create database** menu.
6 Select your database in the **Database** field.
7 Clear the **Create database** and **Create system data** options.
8 Select the option **Import extra data**.
9 In the **Data to import** frame, select **Forms**.
10 Click **Create** to validate the import of the forms.

Creating forms

Use the **Tools/ Reporting/ Forms**menu item to display the list of forms.

Basic information

1 Enter the name of the form.

2 Select the type of form: (list or detail)

Both types can contain text and predefined images.

Differences between the two types:

- List: enables you to print a list of records such as it is displayed in the active list (according to the columns the list contains and the filters applied).
- Detail: enables you to print fields from a record detail (example: the detail of an asset), and linked record lists (example: the components of this asset).

3 Select the main table for the form.

Warning:

The **Table** field (SQL name: TableName) lets AssetCenter propose only those forms relevant to the given list.

Editing forms and objects

To edit a form you define objects and position them on the page.

To insert a new object on the page:

1 Select the Form tab.

2 Click the icon of the object that is on the left-hand side of the page.

Icon	Function
	To select an object in the form in order to modify it, for example.
	To add fixed text and variables independent of the records being printed (current date, for example).
	To add an image.
	To insert a formula containing field values and fixed text strings.

Icon	Function
	**To select an object in the form in order to modify it, for example.**
	To insert a list of records. This tools enables you to position the list on the page. For detail forms, it also enables you to define the linked table containing the records and list of fields to be printed.

3 Position the mouse cursor on this page.
4 Click the left mouse button.
5 Trace a frame with the mouse: This frame defines the space reserved for the object.
6 Double-click the object's position. A window describing the object's properties is displayed.
7 Define the properties of the object.
8 Click **Modify**.

You can insert the following objects in a form:

Fixed text

This concerns text that is independent of the records being printed. You can combine any kind of character, as well as variables:

$D: Date printed.

$U: Name of the AssetCenter user who is printing the form

$C: Page number.

$N: Total number of pages printed.

 Warning:

Do not put quotes around the text.

Example:

```
Document printed on $D on $U
```

Formulas

Formulas are only available in "detail" type forms.

Formulas bring together:

- Field values from the AssetCenter database.
- Fixed text surrounded by quotes.

Example:

```
"Asset:" CodInt " / " Brand
```

Formulas do not permit you to perform calculations.

Lists

- "List" type forms: You can only include one list. This list will be replaced by the list in the active window when you select the **File/ Print** menu item.
- "Detail" type forms: The number of lists is not limited. Lists display all records related to the current record. For example: All the components of an asset.

Images

You can insert images (logos, etc.).

Note:

When you draw up a "list" type form you cannot select the fields to print. AssetCenter prints the fields that appear in form of columns in the lists.

When you draw up a Detail form, you can select the fields to be printed one by one.

Properties of objects in forms

Position and size

To modify the position and size of an object, simply drag the object or its edges.

Note:

You can move and re-size several objects at once by selecting them simultaneously. If you draw a selection frame around several objects with the mouse, all objects in the frame will be selected; or select the objects one by one, while holding down the CTRL key.

Properties

If you double-click an object, or use the **Internal forms/ Properties** menu item, a list of the selected object's properties is displayed.

The property list has two columns: The first contains property names and the second lets you edit the property values.

To modify a property, click in the second column with the left mouse button.

Simple properties may be edited directly (text, formulas, lists, background colors, text colors, text alignment, object alignment.) Complex properties (font for the text or formula, frame, list contents, image) display additional windows.

Text

Enter your text directly.

Formula

To help you in creating formulas, you can display a drop down list with a tree that displays the fields that are accessible and compatible with the type of form selected. Clicking a node in the tree replaces the current selection with the field you selected in the list. You can insert text between fields as long as you include quotes.

Background color, text color

You may choose these colors from 16 available colors.

Text alignment

This determines how text is aligned within the frame. A drop-down list displays the different kinds of alignment (Centered, Left aligned, Right aligned.)

Alignment on the page

This determines the text's horizontal alignment in the page. A pop-up window lets you select the alignment:

- **Left aligned**.
- **Right aligned**.
- **Centered**.
- **Relative alignment**: In this case, the object keeps the position you defined in the form creation screen.

Font

To select the character set of the object and its size, click next to cell being edited.

Frame

To add a border around objects, click the button in the edit cell: A configuration screen for the border is displayed.

The **3D** style gives an embossed effect to the frame.

For borders without the 3D style, you can select which borders will be displayed, their color and thickness.

Image

To insert an image:

1 Click the button in the edit cell.

2 Select the graphic file in the "Select image" screen.

Link in list

To select a list to be displayed in the form, use the drop down list. Click the list you want (for example, the list of assets used by an employee).

List contents

Note:

You can only configure the contents of the list when editing a detail form.

To define the contents of a list, click the button in the edit cell. A configuration window is displayed. All columns in the list are shown.

This screen lets you define:

- the title of each column,
- the formula used to define the contents of the column,
- the size of the column, (percentage used by column)
- the alignment for the title and contents of each column,
- the alignment of each title or contents of each column,
- the vertical and horizontal separators.

To remove a column from the list, use the "Delete" key.

To insert a column in the list, edit the last line of the list.

Each cell can edited using the same principles as for the list of properties.

Design grid

The design grid is made up of vertical and horizontal lines that cover the background of the screen.

The **Form/ Grid** contextual menu item allows you to:

- Show and hide the design grid.
- Define the spacing between lines in the grid.

Only the points of intersection between horizontal and vertical lines are shown. The spacing between lines determines the precision with which you can position objects on the page.

Form page setup

The **Form/ Page setup** contextual menu allows you to define:

- The page format
- **Portrait** or **Landscape** layout
- Document margins
- Header and footer

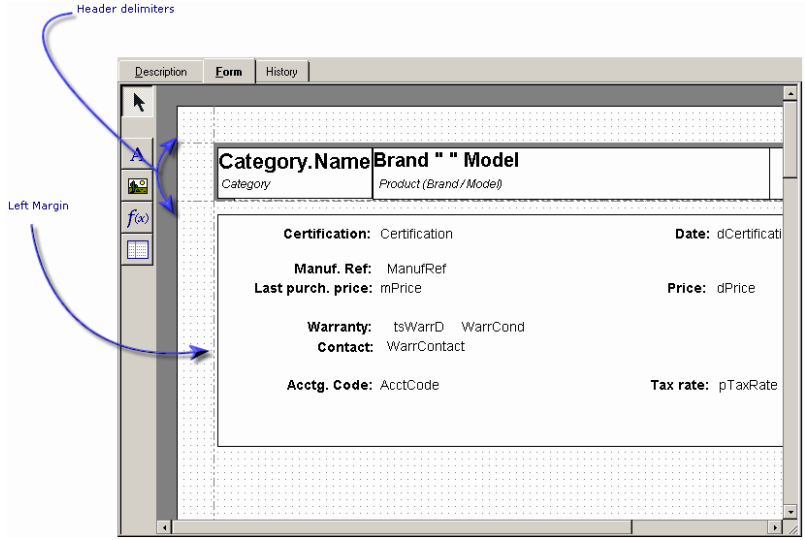
Note:

Document margins, footer and header zones can be directly modified in the edit zone. Simply drag the margin marker and zone limits for the header and footer, which are shown in dotted lines.

To insert text in page headers and footers:

1 Select the **Form/ Page setup** menu item.
2 Check the **Header** and **Footer** boxes.
3 Click **OK**.
4 Go to the header or footer zone in the form (header and footer zones are shown with horizontal dotted lines).
5 Insert objects here or move objects from the main page zone.
6 Confirm your modifications by clicking **Modify**.

Figure 4.1. Forms - header zone



Note:

You cannot move objects from the header and footer zones to the main page zone.

How to easily create regular reports

To produce reports you need regularly, we recommend that you:

1 create a "view" with the appropriate parameters.

2 associate this view with a form.

The view lets you define:

- The sort criteria.
- The filter to apply and the filter values.
- The list of visible columns.

The form lets you organize the report page layout.

To print the report:

1 Display the view you created beforehand (**Tools/ Views** menu item.)

2 Print from the displayed view (**File/ Print** menu item.) Make sure you select the correct "Type" of report and the appropriate "Form".

Identifying forms specific to a given module

To identify the forms specific to a given module:

1 Launch AssetCenter.

2 Display the list of forms (**Tools/ Reporting/ Forms**).

3 Right-click in the list of the window that opens.

4 Select **Configure list** in the shortcut menu that appears.

5 Add the **Domain** link (Domain) to the columns in the list.

6 Click **OK**.

7 Sort the list from the **Domain** column.

8 The forms for a module are identified by the domain name.

Example: **/Procurement/Forms/**.

Associating a form with a button in a screen

There are several ways to associate a form with a button in a screen.

For more information, see:

- The **User interface** guide, chapter **Customizing a client workstation**, section **Customizing buttons**.
- The **Administration** guide, chapter **Customizing the database**, section **Customizing existing objects/ Customizing objects / Customizing a detail/ Creating action buttons**.
- The **Administration** guide, **Customizing the database**, section **Creating new objects/ Creating action buttons**.

5 Crystal Reports

CHAPTER

This chapter explains how to print reports with AssetCenter.

Use the Tools/ Reporting/ Reports menu item to display the list of reports

Note:

The Crystal Reports are not available for the Unix version of AssetCenter.

Installing and using the reporting tool

General overview

AssetCenter makes use of Crystal Reports reporting software. These reports have the file extension **.rpt**.

Note:

The commented list of available reports is available in the **Reports.txt** file of the **datakit\standard\reports** sub-folder, located in the AssetCenter installation folder.

The Crystal reports are stored in the **\datakit\standard\reports\rpt** folder.

Do not modify the structure of the **Reports.txt** file since the import script of the reports uses it. On the other hand, if you only want to import a selection of reports into your working database, you can delete **full** lines from this file before executing the import script. You may also add your own reports.

You do not need to have Crystal Reports to print existing reports

A limited version of Crystal Reports is installed with AssetCenter if you check the appropriate option during the installation.

This limited version is sufficient to preview and print existing reports with the current data from the AssetCenter database.

You need to have Crystal Reports to modify existing reports or to create new ones

AssetCenter obviously cannot let you create these reports directly.

To do this, however, you need to acquire the full version of Crystal Reports. The constraints to be taken into consideration are described below:

Versions of Crystal Reports supported	Versions 7.x and 8.0 Japanese version: 7.0
Languages of Crystal Reports supported	• French • English • German • Spanish • Italian • Japanese • Polis You may use different languages for Crystal Reports and AssetCenter.

Installing the full or runtime version of Crystal Reports

AssetCenter's installation program lets you install a limited version of Crystal Reports. This can be used to print the existing reports. For this, you need to select the appropriate option when installing AssetCenter.

If you have already installed, or if you plan to install a full version of Crystal Reports, you do not need to install the limited version supplied with AssetCenter.

Installing preconfigured Crystal Reports in your database

AssetCenter is supplied with preconfigured reports. These are already installed in the demonstration database, however, you still need to insert them in your working database.

Note:

The preconfigured reports are not provided with the following versions of AssetCenter:

- Japanese version
- Polish version

To insert the reports one by one:

1. Launch AssetCenter.
2. Open your working database.
3. Use the **Tools/ Reporting/ Reports** menu item.
4. Create a new report.
5. Select the **File** tab in the detail of the report.
6. Click **Import**.
7. Select the **.rpt** extension file that corresponds to your needs in the **\datakit\standard\reports\rpt** sub-folder of the AssetCenter installation folder.

Importing reports during database creation

To import reports during the database creation:

1. Launch AssetCenter Database Administrator.
2. Select the **File/ Open** menu.
3. Select the **Open database description file - create new database** option.
4. Select the **gbbase.xml** file, located in the **config** sub-folder of the AssetCenter installation folder.
5. Select the **Action/ Create database** menu.
6. Populate all the fields that enable you to create a database. Consult the **Administration** guide, chapter **Creating a AssetCenter database** to find out what fields are necessary.
7. Select the option **Import extra data**.
8. In the **Data to import** frame, select **Crystal Reports**.
9. Click **Create** to validate the creation of the database.

Importing reports in an existing database

To import reports into an existing database:

1. Launch AssetCenter Database Administrator.
2. Select the **File/ Open** menu.
3. Select the **Open database description file - create new database** option.
4. Select the **gbbase.xml** file, located in the **config** sub-folder of the AssetCenter installation folder.
5. Select the **Action/ Create database** menu.
6. Select your database in the **Database** field.
7. Clear the **Create database** and **Create system data** options.
8. Select the option **Import extra data**.
9. In the **Data to import** frame, select **Crystal Reports**.
10. Click **Create** to validate the import of the reports

Detail of a report

Use the **File/ Reports** menu item to display the list of reports.

A report detail in AssetCenter is made up of the following information:

File

You cannot edit this field directly. It indicates the name of the report file (with its extension and the relative path of its folder) that was imported using the **Import**.

The following buttons let you work with reports:

- **Import**: This button in the report detail enables you to import (the first time to create the report, following times to modify the report) an external report. External reports have the **.rpt** file extension. Importing an external report updates the **File** field (SQL name: FileName) in the AssetCenter report detail.
- **Export**: This button in the report detail enables you to create an **.rpt** file from a report contained in the AssetCenter database. By default, the dialog box that opens proposes the name of the file contained in the **File** field. You can modify this. Doing this allows you to modify a report using the external reporting program.
- **Preview**: This button, accessible via the **File/ Print** menu item, lets you preview the report on-screen before printing it.
- **Print**: This button, accessible via the **File/ Print** menu item, lets you print the report.

Note:

When you press the **Preview** or **Print** buttons, AssetCenter creates a temporary file from the report in the database. This file is processed by the Crystal Reports print engine. The temporary file is erased immediately afterwards. The data displayed or printed is the data currently in the database.

How to modify a Seagate Crystal report

In order to modify a report contained in the AssetCenter database, you need to have Crystal Reports.

Use the following procedure:

1. Display the detail of the report using the **File/ Reports** menu item.
2. Click the **Export** button to create a **.rpt** file.
3. Modify the **.rpt** report using Crystal Reports and save it.

4 Display the report detail again using the **Tools/ Reporting/ Reports** menu item.

5 Import the **.rpt** file to update it and modify the record.

Crystal Reports statistics

To display Crystal reports that are automatically updated, use the **Tools/ Reporting/ Crystal Reports statistics** menu item.

You can display the same reports as when using the **Tools/ Reporting/ Reports** menu item.

Nature

Indicate the nature of the report to be displayed. The field to the right of this field enables you to select a given report. The reports that are available depend on the "Nature" you select.

Automatic refreshing button

This button is represented by the icon.

- Click this button to refresh the report.
- Right-click this button to define the frequency of automatic refreshing of reports.

Zoom button

This button is represented by the icon.

Modifies the zoom factor (3 levels).

Creating a detail report

A "detail report" is a report that prints the detail information on one or more records selected in a list.

Example of utilization

1 Display the list of assets.
2 Select an asset.
3 Select the **File/ Print** menu item.
4 Set the **Type** field to "Detail report (Crystal Reports)".
5 Select the report.
6 Print.

This procedure prints a detail report for each selected record.

Configuring reports under Crystal Reports

To obtain a detail report, follow the procedure below (example taken under Crystal Reports Professional 5.0):

1 Use the **Insert/ Formula Field** menu item to create a formula field. Its name must respect the following syntax:

```
<SQL name of the table for which the report is cotextuall>Id
```

Note:

You must respect the case of the SQL names of tables.

For example, to create a contextual report on the table of assets, the formula is:

```
amAssetId
```

Note:

Do not confuse the syntax of the formula field name with the SQL name of the primary key field. For example, the primary key of the table of assets is "lAstId", which is different from "amAssetId".

The "CurrentUserId" formula (respect the case) makes it possible to identify the user printing the report. When printed, this formula takes the value of the identifier number (i.e. the value of the field with SQL name: "lEmplDeptId" for the current login) of the user connected to the AssetCenter database.

If you want to see the result of the report for a given record in the contextual table, edit the formula field and enter the primary key of the table for an existing record in the AssetCenter database.

For example:

```
512
```

Note:

Edit the formula field in the window, which is automatically displayed when you confirm the name of the new formula field. If the formula field exists already, click the **Edit** button to edit it.

2 Use the **Report/ Edit Selection Formula/ Record** menu item to edit the selection formula. It uses the following syntax:

```
{<SQL name of the context's table>.<SQL name of the field that is a pr
imary key>} = @<Name of the formula's field>}
```

The case used for the SQL names of tables and fields is unimportant. Example:

```
{amAsset.lAstId} = {@amAssetId}
```

Using the above procedure, AssetCenter automatically identifies the report as being contextual when it is imported into the database. You will see this when you perform the following:

1 Access the list of reports using the **File/ Reports** menu item.

2 Create a new report.

3 Import the Crystal Reports file (**.rpt** extension) by clicking the **Import** button.

4 Once this file has been added, you will notice that the **Table** field (SQL name: TableName) shows the SQL name of the context table. If this is not the case, verify the formula field and the selection formula in the Crystal Report.

Printing a report

Detail reports

Note:

To learn more about detail reports and how to create one with Crystal Reports:
→ Creating a detail report [page 84].

To print a detail report in AssetCenter:

1 Display the list of records whose details you want to print (**Portfolio/Assets and batches**, for example).
2 Select at the same time all the records you want to print.
3 Select the **File/Print** menu.
4 Populate the **Type** field with the value **Detail report (Crystal Reports)**.
5 Select the report.
6 Click **Print**.

This procedure prints a detail report for each selected record.

Non-contextual reports

Note:

Non-contextual reports are reports of lists or graphs.

Unlike **detail reports**, they do not use tables as contexts.

To print a non-contextual report in AssetCenter:

1 Select the **File/Print** menu.
2 Populate the **Type** field with the value **Non-contextual report (Crystal Reports)**.
3 Select the report.
4 Click **Print**.

This procedure prints a list or graph report

Identifying Crystal reports specific to a given module

To identify the Crystal reports specific to a given module:

1. Launch AssetCenter.
2. Display the list of reports (**Tools/ Reporting/ Reports**).
3. Right-click in the list of the window that opens.
4. Select **Configure list** in the shortcut menu that appears.
5. Add the **Domain** link **[Domain]** to the columns in the list.
6. Click **OK**.
7. Sort the list from the **Domain** column.
8. The reports for a module are identified by the domain name.

   Example: **/Portfolio management/IT/Reports/**.

Associating a report with a button in a screen

There are several ways to associate a report with a button in a screen.

For more information, see:

- The **User interface** guide, chapter **Customizing a client workstation**, section **Customizing buttons**.
- The **Administration** guide, chapter **Customizing the database**, section **Customizing existing objects/ Customizing objects / Customizing a detail/ Creating action buttons**.
- The **Administration** guide, **Customizing the database**, section **Creating new objects/ Creating action buttons**.

6 Actions

CHAPTER

This chapter explains how to define actions with AssetCenter.

Use the **Tools/ Actions/ Edit** menu item to create actions.

You can execute actions via the Tools/ Actions menu item or the contextual list of "Actions" in the toolbar.

Definition of an action

An action enables you to automate, either completely or partially, the tasks performed on an AssetCenter database.

There are several types of actions:

- Executable
- DDE
- Messaging
- Script: modifying an object in the AssetCenter database
- Wizard
- Printing
- Deployment

- Action

Actions must first be defined in order for them to be executed by selecting them from a list.

Note:

You can define a domain for an action, as well as categorize domains according to their functions using functional domains.

Functional domain

AssetCenter enables you to define domains that group together the functions of the software. By default, certain functional domains are provided with the software: They correspond to the modules that you can activate or deactivate using the File/ Active modules menu item.

Functional domains are used to create and classify the information displayed in the Functions and favorites pane. Thus, once you select a functional domain for an action, the action will appear in the Functions and favorites pane under the heading of that functional domain.

Note:

The contents of the Functions and favorites pane is reorganized and modified according to the context. If your action is contextual (it can't be executed unless a specific screen is open, for example), then it will not appear in the Functions and favorites pane unless the current context corresponds to its definitions (if that specific screen is currently displayed, for example).

To define a functional domain:

1 Click **New**.

2 Select the **Administration/ Functional domains** menu item.

3 Assign a **Name** to your functional domain. This name is the one that will appear in the Functions and favorites pane. By default, AssetCenter assigns an **SQL name** to the functional domain; you can modify this value if you want.

4 You can select a **Parent domain** for the functional domain as well if you want.

5 Validate your creation by clicking **Create**.

Warning:

If access to a functional domain is fully forbidden (read and write) for a user, they will not be able to access actions, vues or reports in this domain. In practice, it recommended to authorize read access to the following tables when defining a functional domain:

- amFuncDomain
- amViewDef
- amReport
- amForm
- amAction
- amScriptLibrary

Creating an action

This section describes how to create an action:

- Types of actions
- General method
- Populating the DDE tab
- Populating the Messaging tab

Types of actions

AssetCenter enables you to define several types of actions.

Note:

AssetCenter only allows you to create **Executable**, **Messaging** or **Printing** type actions. **DDE**, **Script** and **Wizard** type actions are reserved and can be executed only.

Executable actions

An **Executable** action causes a program to be executed.

It launches an **.exe**, **.com**, **.bat**, or **.pif** application. You can also refer to any type of document, as long as its extension is associated with an application in the file manager.

DDE actions

A **DDE** action sends a DDE request to a DDE server application (or DDE compliant application) capable of handling DDE requests.

DDE stands for "Dynamic Data Exchange"; it designates a method for dynamically exchanging information between programs. AssetCenter uses DDE have commands executed by another application.

Example: Through DDE, you can request Microsoft Word to open a file whose name is specified and with given contents.

Messaging actions

Messaging actions allow you to send a message:

- Via AssetCenter's internal messaging system.
- Via an external VIM standard messaging system (Lotus Notes, Lotus cc:Mail, etc.).
- Via an external MAPI standard messaging system (Microsoft Exchange, Microsoft Outlook, etc.).
- Via an external SMTP standard messaging system.

Warning:

You can only send messages via those messaging systems that you are able to connect to.

To issue a VIM, MAPI or SMTP standard message, AssetCenter uses:

- The **Account** (SQL name: MailLogin) and **Password** (SQL name: MailPassword) fields of the **Messaging** tab of the detail of the employee who opened the AssetCenter database (table of departments and employees) to identify the sender of the message,
- The **EMail** field (SQL name: EMail) in the **General** tab of the employee detail to identify the recipient of the message.

To send a message via AssetCenter's internal messaging system, AssetCenter uses the **Login** and **Password** fields in the **Profile** tab of the details of both the sender and recipient.

Note:

The internal messaging address of an AssetCenter user is the same as the **Login**.

Warning:

The administrator must create a user with the name "Admin" and populate the **Account**, **EMail** and **Password** fields in order to use an external messaging system and make sure that AssetCenter Server functions correctly.

Script actions

Script actions enable you to perform any operation on an AssetCenter database. They give advanced users extensive control over the database, allowing them to perform operations that cannot be performed with other types of actions, and in particular:

- Creating records
- Deleting records
- Duplicating records
- Modifying one or more objects in the AssetCenter database, e.g. all the records in a table, a field or a link.

The operations performed by this type of action are described by a Basic script; this enables you to use complex functions similar to those in the AssetCenter APIs.

Note:

The complexity of the usable functions in **Script** action, associated with the ability to make in-depth changes to the database, make this type of action potentially dangerous to the database integrity. Therefore it should be used by advanced users only.

Different functions are used to change the value of a database object depending on the context of the action:

- If the action has no context, you must use functions derived from AssetCenter APIs such as **AmSetFieldStrValue()** or **AmSetFieldLongValue()**.

- If the action has a table as a context, you can use the **Set()** function; it has the following syntax:

```
Set [<Link.Link.Field>]=<Value>
```

Wizard actions

Wizards are intended to guide you step by step though complex or recurrent tasks in AssetCenter. Wizards are designed via a dedicated programming language.

Note:

Wizards are complex actions. For further information, refer to the **Wizards** chapter of this guide.

Printing-type action

A **Printing** type action enables you to print a report or form.

You need to populate the following fields for this type of action:

- **Type** field (seFormType): enables you to indicate the type of document to print (report or form).
- **Report** or **Form** fields: enables you to indicate which report or form must be used;

Note:

The context of the action is determined by the context defined for the report or the form.

Deployment-type action

A **Deployment** type action enables you to launch a deployment on a list of computers defined in the Computers table.

This action is contextual in the Computers table.

You need to populate the following fields for this type of action:

- **Deployment workflow**: select the deployment workflow that you created or imported.
- **Deployment server**: select your deployment application server.

For more information about implementing and configuring the deployment, refer to the **Desktop Administration** guide.

Action-type action

An **Action** type action enables you to execute another action according to the current request.

You need to populate the following fields for this type of action:

- **Action to execute** field: enables you to specify which action must be executed.
- **Selection query** field: enables you to enter the query script that specifies the context of the action application.

General method

To create an action:

1 Select the **Tools/ Actions/ Edit** menu.

2 Click **New.**

3 Enter a name for the action.

4 In the **Type** field (SQL name: seActionType), specify the type of action you want to create. The type of action you select controls the display of one of the following tabs:

- **Executable**
- **DDE**
- **Messaging**
- **Script**
- **Wizard**
- **Printing**

5 You can populate the **SQL name** field (SQL name: SQLName) of the action detail, but it is not required. The SQL name is a unique way to identify the action; it is used in particular when executing an action via a DDE command (in cases where AssetCenter is used as a DDE command server).

Note:

If you do not populate the **SQL name** field, AssetCenter does so automatically by generating a standard SQL name.

6 Populate the **Context** field (SQL name: ContextTable):
   - If you select a table from the drop-down list, the action is context-sensitive: It will only be proposed if you display the list of records in that table, or the details of one of those records.
   - If the action does not depend on a table, select the **(No table)** option at the top of the drop-down list.

7 Populate the **Domain** field, which enables you to specify the functional domain to which the action belongs. The action will appear under this domain in the Functions and Favorites pane.

8 You can attach an icon to the action, but it is not required:

   To do this, use the square located at the top left of the action detail screen. The image will then appear in the "Actions" context-sensitive list in the toolbar. The active icon in the list (the one displayed on the screen by default) is the icon for the last action executed using the toolbar.

9 Populate the fields in the **Description** tab, and the fields in the specific tab for the "Type" of action you want to create.

10 Click **Create**.

Note:

The AssetCenter administrator sees all actions, whether or not they are shared, and whether or not they were created by the administrator.

Populating the DDE tab

The information concerning a particular DDE action is located in the DDE tab of the action detail.

This tab is displayed only if you assigned the value DDE to the **Type** field (SQL name: seActionType) field in the basic information for the action.

The DDE mechanisms are based on the "services" provided by the software. In order to execute DDE mechanisms, you must define a "topic" that indicates the context in which the "commands" should be executed.

Therefore you must indicate:

- In the **Service** field (SQL name: DDEService), the name of the DDE service provided by the executable you want to call. This is usually a unique service for an executable. Refer to the documentation of the executable for the list of services is provides.

- In the **Topic** field (SQL name: DDETopic), the context in which the action should be executed.
- In the **Command** field (SQL name: DDECommand), the commands you want the external application to execute.

  For Word, the command may be a Word Basic or a Visual Basic command.

  If the DDE service of the called application allows it, you can put several commands side by side.

  You must follow the syntax required by the external application.
- If the service is not present, indicate in the **File** field (SQL name: ActionFile) the file used to start the application that activates the service. This is the main application that responds to DDE commands.

Important note

Commands sent to the external application are surrounded by square brackets ("[", "]". For example (using Microsoft Word):

```
[FileOpen("c:\tmp\test.txt")]
```

- When the action is contextual, you can use variables to reference the value of a field in the database. Since these variables are also surrounded by square brackets, AssetCenter cannot differentiate between commands and variables by itself. You must identify commands by prefixing the square brackets with a backslash character "\". Thus the previous example is written (in the case of a contextual action):

  ```
  \[FileOpen("c:\tmp\test.txt")\]
  ```

  You can combine commands and variables, as shown below (in this case the context is the table of assets):

  ```
  \[FileOpen("c:\tmp\"+"[AssetTag]"+".txt")\]\[FileClose()\]\[FileExit()\]
  ```
- If the action is not contextual, the problem does not arise. Text surrounded by square brackets is still considered as commands to send to the external application.

Populating the Messaging tab

Information concerning a Messaging action is located in the Messaging tab of the action detail.

This tab is displayed only if you set to the **Type** field in the basic information of the action to **Messaging**.

Warning:

In order for the system to function correctly, your system's PATH variable must include the folder containing the VIM DLL (VIM.DLL) and MAPI (MAPI.DLL).

What is the Referenced object field used for?

This field is used to select a link from the table selected in the **Context** field.

This field is only used for messages sent via AssetCenter's internal messaging system. It enables you to directly access the object that triggered the issuing of the message by simply clicking the **Referenced object** button in the message detail. When the referenced object is directly the record that triggers the action, you do not fill in the **Referenced object** field (SQL name: RefObject).

How to receive an acknowledgement

If you want the message sender to receive an acknowledgement via their usual messaging service, check the **Acknowledgment** box (SQL name: bAcknowledgment).

This acknowledgement will be sent to the address specified by the **EMail** field (SQL name: EMail) in the **General** tab of the employee who opened the AssetCenter database (in the departments and employees table).

Note:

You cannot receive an acknowledgement for a message sent via the AssetCenter internal messaging system, or via a MAPI or SMTP messaging system.

How to indicate an address

Here are the different ways of indicating an address:

Address with form <Messaging engine>:<Messaging address>

<Messaging engine> can be:

- AM: to force the utilization of AssetCenter's internal messaging system.
- MAPI: to force the utilization of a MAPI standard messaging system (Internet Mail, Microsoft Outlook, etc).
- VIM: to force the utilization of a VIM standard messaging system (Lotus Notes, etc.).
- SMTP: to force the utilization of a SMTP standard messaging system (Internet standard).

<Messaging address> has the usual form corresponding to the messaging system selected. Internal messaging addresses are the same as the "Logins".

Examples of addresses:

- AM:Admin
- MAPI:CathyBernard@taltek.com
- VIM:Cathy Bernard / TALTEK
- SMTP:cbernard@taltek.com

Address with form <AssetCenter login>

In this case, the messaging system used will be that which is indicated in the **EMail** (SQL name: EMail) of the **General** tab of the detail of the employee whose **Login** (**Profile** tab in the detail of the employee) is specified in the address.

If the **EMail** field is not populated, the message is sent via the internal messaging system.

For example:

1. A message is sent to the following AssetCenter logins: "Cathy", "Gerald" and "Philip".
2. The **EMail** fields show "MAPI:CathyBernard@taltek.com" for "Cathy" and "VIM:Gerald Colombo / Taltek" for "Gerald". The **EMail** field of "Philip" is empty.
3. If the sender has a MAPI account, the message will be sent to "Cathy" via MAPI and to the two other recipients via AssetCenter's internal messaging system.
4. If the sender has a VIM account, the message will be sent to "Gerald" via VIM and to the two other recipients via AssetCenter's internal messaging system.

Address with contextual variables

If the action is contextual, you can use variables between brackets [ ].These variables call on values of fields in the AssetCenter database.

For example: To send a message to the user of an asset selected in the table of assets, you can use **[User.Email]**.

Examples of actions

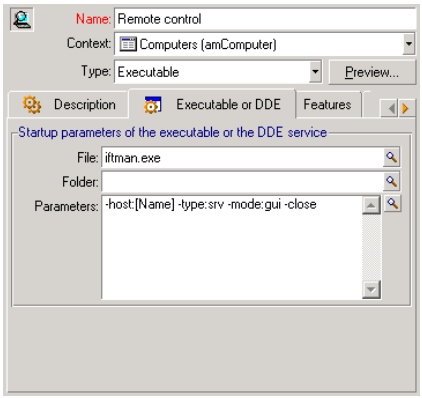
This section provides examples of AssetCenter actions:

- Example of Executable type action.
- Example of DDE type action.
- Example of Messaging type action.
- Example of a Script type action.

Example of an executable-type action.

The following screen describes a non-contextual action that launches AssetCenter Server and connects to a database **acdemo**:

Figure 6.1. Executable-type action - detail window



Example of a DDE-type action

There are numerous applications of **DDE** type actions:

- Inserting AssetCenter data into a Microsoft Excel worksheet.
- Inserting information related to a purchase order in an accounting software package.
- Automatic sending of a confirmation message by fax on closing a ticket.
- Automatic sending of a work order request.
- Etc.

This section describes a simple **DDE** action.

Aim of the action

This action sends confirmation of a purchase request.

This action is triggered from the detail of a purchase request.

The action uses a DDE link between Microsoft Word 7 and AssetCenter. It inserts information on the request in a Word document (details of the person to contract and the request number) and prints it.

Preparations: Creating the letter under Word

You must first create the Word document **LetterTemplate.doc** that will be printed.

This letter will be structured as follows:

Figure 6.2. "LetterTemplate.Doc"

The **LetterTemplate.doc** document can be found in the installation folder of AssetCenter.

The corresponding Word template **Normal.dot** is attached to **LetterTemplate.doc**. It contains a macro, **mymacro.bas**:

```
Attribute VB_Name = "MyMacro"
Sub PrintLetterTemplate(MrMrs, FirstName, Name, Adr1, Adr2, Zip, City, Re
qNo)
'
' PrintLetterType Macro
'
    Application.WindowState = wdWindowStateMinimize 'Run Winword in the b
ack end
    Documents.Open ("LetterType.doc") 'Open letter pattern
    Documents("LetterType.doc").Activate

    Selection.Find.ClearFormatting 'Clear parameters for Find function
    Selection.Find.Replacement.ClearFormatting 'Clear parameters for Repl
ace function

    With Selection.Find
        .Text = "<MrMrs>"
        .Replacement.Text = MrMrs
        .Forward = True
        .Wrap = wdFindContinue
        .Format = False
        .MatchCase = False
        .MatchWholeWord = False
        .MatchWildcards = False
        .MatchSoundsLike = False
        .MatchAllWordForms = False
    End With
    Selection.Find.Execute Replace:=wdReplaceAll 'Execute replacement

    With Selection.Find
        .Text = "<FirstName>"
        .Replacement.Text = FirstName
    End With
    Selection.Find.Execute Replace:=wdReplaceAll

    With Selection.Find
        .Text = "<Name>"
        .Replacement.Text = Name
    End With
    Selection.Find.Execute Replace:=wdReplaceAll

    With Selection.Find
        .Text = "<Adr1>"
        .Replacement.Text = Adr1
    End With
    Selection.Find.Execute Replace:=wdReplaceAll
```

```
    With Selection.Find
        .Text = "<Adr2>"
        .Replacement.Text = Adr2
    End With
    Selection.Find.Execute Replace:=wdReplaceAll

    With Selection.Find
        .Text = "<Zip>"
        .Replacement.Text = Zip
    End With
    Selection.Find.Execute Replace:=wdReplaceAll

    With Selection.Find
        .Text = "<City>"
        .Replacement.Text = City
    End With
    Selection.Find.Execute Replace:=wdReplaceAll

    With Selection.Find
        .Text = "<ReqNo>"
        .Replacement.Text = ReqNo
    End With
    Selection.Find.Execute Replace:=wdReplaceAll

    ActiveDocument.PrintOut 'Print document
    ActiveWindow.Close (wdDoNotSaveChanges) 'Close document w/o updating
    MsgBox ("Your document is being printed.") 'Notify user
End Sub
```

Step 1: Creating the AssetCenter action

To create the AssetCenter action:

1. Open the table of actions (**Tools/ Actions/ Edit** menu item).
2. Click **New** to create a new action.
3. Populate the **Context** field (SQL name: ContextTable) in order for it to show the table of purchase requests.
4. Set the "Type" field to **DDE**.
5. Populate the **DDE** tab in order to define the DDE link between AssetCenter and Microsoft Word 7.

In the DDE tab:

1. The **Service** field (SQL name: DDEService) is set to "Winword".
2. The **Topic** field (SQL name: DDETopic) is set to "System".
3. The **Service start parameters** frame shows **Winword.exe** and its path.
4. The **Command** field (SQL name: DDECommand) specifies the macro to be launched and its parameters:

```
\[MyMacro.PrintLetterTemplate "[Requester.MrMrs]", "[Requester.FirstName]
", "[Requester.Name]", "[Requester.Location.Address1]", "[Requester.Locat
ion.Address2]", "[Requester.Location.ZIP]", "[Requester.Location.City]",
"[ReqNumber]"\]
```

Click **Create** to confirm the creation of the action.

Step 2: Launching the action

To launch the action:

1 Open the table of purchase requests.

2 Select a purchase request.

3 Launch the action via the **Tools/ Actions** menu item.

When the action is launched:

1 Microsoft Word is launched and loads **LetterTemplate.doc**.

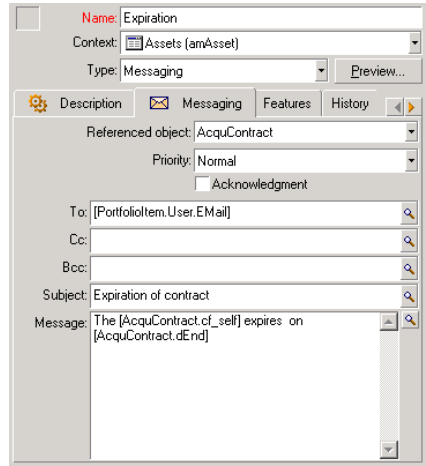
2 The details of the person to contact and the request number are inserted in the letter.

3 The letter is printed.

Example of Messaging type action

You send a message from the list of assets to indicate the expiration date of an asset's lease schedule to the user of that asset. This asset must have Lease listed as its mode of acquisition and must be linked to the lease schedule (Acquis. tab). In order for the referenced object to be the purchase request and not the request line, configure the action detail as follows:

Figure 6.3. Messaging-type action with a referenced object - detail window



Example of a Script type action

Creating a **Script** type action basically involves writing the Basic script that modifies the AssetCenter database.

Note:

The use of functions specific to these actions is authorized within these scripts. These functions are indexed in the manual entitled "Programmer's Reference", chapter "Index of functions by field of application", section "Built-in functions"

Foreword

To prepare for creating the action, follow these steps:

1. Select the **Tools/ Actions/ Edit** menu item and click the **New** button in the action detail screen.
2. Assign a name to the action you are going to create, e.g. "Test", and set the **Type** field (SQL name: seActionType) to **Script**. Do not select a context for the action. Click **Create**.
3. In the **Script** tab, click the button to display the script builder window. The programmable function, called **Success()**, used for these actions does

not require an explicit return code. In the following example, we will create a new record in the Natures table based on the information contained in the table below:

Field label	SQL name of the field	Value of the field
Name	Name	Mini-computer
Created	seBasis	Portfolio item
Can be connected	bIsCnxClient	This box is checked

Writing the script

Enter the following example:

```
Dim lrec As Long
Dim lres As Long
  lrec=AmCreateRecord("amNature")
  lres=AmSetFieldStrValue(lrec, "Name", "Mini-ordinateur")
  lres=AmSetFieldStrValue(lrec, "seBasis", 1)
  lres=AmSetFieldStrValue(lrec, "bIsCnxClient", 1)
  AmInsertRecord(lrec)
```

Note:

This action creates the desired nature without any user intervention.

Demonstration of the "Set()" function

Now we will create the same nature from a **Script** type action, by specifying the Natures table as the context for the action. In this case, we write the script as follows:

```
Set [Name]="Mini-ordinateur"
Set [seBasis]=1
Set [bIsCnxClient]=1
```

Note:

To execute this action, the user must open the Natures table and click **New**. After executing the script, the user must also click **Create** to validate the creation.

Tip

If you want to invalidate the execution of an action within a script, simply set the value of the return code to something other than 0 (12001, for example). This value is considered as an error code. The next command interrupts the action and cancels all changes already made:

```
RetVal=12001
```

Using variables

In the **Executable**, **DDE**, or **Messaging** tabs of the detail of a contextual action, you can use variables that reference the contents of fields, features or calculated fields in the database.

They use the form **[Link.Link.Field]**.

For help in entering these variables, click the magnifier to the right of the field to be populated.

Everything not contained within braces [] is considered as text.

For example: **[Link.Link.Field].doc**, calls the value of the field **Field** in the table linked to the main table going through the links **Link.Link**.

Warning:

In order for the variables to work, the **Context** field of the action detail must show a table in AssetCenter and you must select a record in the list of records of the table before executing the action.

Specificities of the Sybase SQL Anywhere database engine

When using Sybase SQL Anywhere as AssetCenter's database engine, it is not possible to write "{d" or "m" at the start of a field in an action.

If you need a field in an action to start with "{d" or "m", we recommend preceding these strings by a space.

Testing an action

To test an action when creating it, use the **Preview** found in the top-right corner of the detail of the action to be tested.

Calculate button

Once the context is selected, click the **Calculate** button. This fills in the fields in the **Executable**, **DDE**, or **Messaging** tabs. Check that the variables have been correctly extracted from the record selected in the **Context** field (SQL name: ContextTable).

Execute button

This button enables you to execute the action directly from this screen.

Executing an action

You may execute an action in one of several ways:

- Using the drop-down list in the toolbar:
  - The button is replaced by the icon associated with the last action used on this workstation, if this icon exists. If an action has already been executed, click the icon, or the icon replacing it to set it off again.
  - The button displays the list of available actions.
  - To insert this pop-up list in the toolbar, use the Tools/ Customize toolbar menu item: it is part of the "Tools" category.
- Using the Tools/ Actions menu item: Click the desired action in the sub-menu.
- From the **Preview** button found in the top-right corner of the detail of the action:
  - If the action is contextual, specify the Context (SQL name: ContextTable) by selecting a record in the action's reference table.
  - Click **Execute** to execute the action.

- From the shortcut menu (accessible by right-clicking). If at least one action is available for the open table, the Actions entry is shown in the shortcut menu.

Multiple-selection in lists

You may select several records in a list and apply an action to them.

In this way, you can select several assets and send the same message to their users.

Wizard-type actions

Wizards are composed of a succession of pages. Each of these pages displays information or requires user input, such as a selection to be made or data items to be entered.

Navigating between the different pages of a wizard is simple:

- Once the page is populated appropriately, you can move to the following page (determined by a transition) by clicking the **Next** button. This button is not available for the final page.
- You can always go back to make any corrections by clicking the **Previous** button.
- You can execute the final action of the wizard at any given moment by clicking the **Finish** button. If the wizard does not have sufficient information in order to perform its designated task, the appropriate page is displayed.

You can cancel the execution of a wizard completely (and as a consequence, its associated action) by clicking the button **Cancel**.

Associating an action with a button in a screen

There are several ways to associate an action with a button in a screen.

For more information, see:

- The **User interface** guide, chapter **Customizing a client workstation**, section **Customizing buttons**.

- The **Administration** guide, chapter **Customizing the database**, section **Customizing existing objects/ Customizing objects / Customizing a detail/ Creating action buttons**.
- The **Administration** guide, **Customizing the database**, section **Creating new objects/ Creating action buttons**.

CHAPTER 7

Messaging

AssetCenter gives you the possibility of managing two types of messages:

- Messages issued from AssetCenter and sent to the AssetCenter database via its internal messaging system.
- Messages created in AssetCenter and sent via an external messaging system.

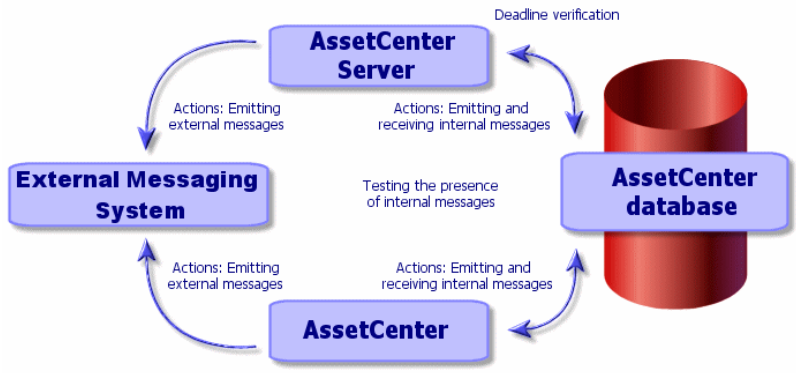
Overview of messaging

AssetCenter manages message sending by using the following protocol types:

- AM (AssetCenter)
- SMTP
- MAPI
- VIM

When receiving messages, AssetCenter only manages those messages using the AM (AssetCenter) protocol.

Figure 7.1. Messaging - overview



How to issue messages

Messages are generated via **Messaging** actions.

These actions must first be created in order for a message to be able to be sent.

The action is triggered in different ways:

- Manually, by selecting the action in the list given by the **Tools/ Actions** menu item.
- Automatically, via AssetCenter Server.
- Automatically, via AssetCenter.

The creation of messaging-type actions is described the section Populating the Messaging tab [page 97].

How to consult messages

Consulting messages sent to the internal messaging system

An agent test for the arrival of new internal messages and informs the users of the presence of new messages.

These messages can be consulted:

- Using the **Tools/ Messages** menu item.
- From the message box which informs of the presence of new messages.

Consulting messages sent to an external messaging system

The recipient of these messages can consult them in the usual way under the external messaging system client.

Acknowledgement of receipt

 Note:

You cannot receive an acknowledgement for a message sent via the AssetCenter internal messaging system, or via a MAPI or SMTP messaging system.

To receive acknowledgment, check the **Acknowledgment** box (SQL name: bAcknowledgment) field in the detail of the **Messaging** type action.

This acknowledgement will be sent to the address specified by the **EMail** field (SQL name: EMail) in the **General** tab of the employee who opened the AssetCenter database (in the Departments and Employees table).

8 Workflow

CHAPTER

This chapter describes in detail how to define and manage workflow schemes.

Definitions

This section defines several key terms used in workflow:

- Workflow
- Workflow activity
- Workflow event
- Workflow transition
- Workflow task
- Workflow activity assignee
- Definition of a workflow execution group

Workflow

Workflow is concerned with the formalization and/or automation of business procedures.

For example, the following processes can be modeled and automated using workflow methods:

- Purchase-request approval procedures.
- Asset moves.
- Etc.

AssetCenter makes it possible for you to define workflow schemes and manage their progress.

Workflow scheme

Creating a workflow scheme in AssetCenter consists of defining:

- Activities.
- Trigger events (events resulting from activities that make it possible to activate transitions).
- Trigger transitions (transitions that trigger activities).
- A context.
- Time limits and alarms.

Figure 8.1. Workflow - simplified scheme

Use the Tools/ Workflow/ Workflow schemes menu item to access the table of workflow schemes (SQL name "amWfScheme").

Workflow instance

In this document, a "Workflow instance" refers to a defined workflow scheme that is being executed.

Workflow activity

A workflow consists of:

- A task to be executed. This task may necessitate user interaction or be carried out automatically by AssetCenter Server.
- Events that trigger transitions to other activities.

Workflow activities are stored in the table of workflow activities (SQL name: amWfActivity).

Use the graphical editor in the Activities tab of a workflow detail to access the detail of a workflow activity.

Workflow event

Workflow events results from activities. They in turn make it possible to activate transitions that trigger other activities.

Events belonging to a workflow scheme are stored in the table of workflow events (SQL name: amWfEvent).

When these events occur, they are recorded in the table of elementary events of a workflow instance (SQL name: amWfOccurEvent).

Use the graphical editor in the Activities tab of a workflow detail to access the detail of workflow event.

Workflow transition

A workflow transition makes it possible to go from one activity to another. They are triggered by an event.

An event can be associated with several transitions.

Transitions that belong to a workflow scheme are stored in the table of workflow transitions (SQL name: amWfTransition).

Use the graphical editor in the Activities tab of a workflow detail to access the detail of workflow transitions.

Workflow task

A workflow task is an assigned task to be carried out, resulting from the triggering of an activity.

In order for workflow tasks to be recorded in the table of workflow tasks (SQL name: amWfWorkItem), the **Log task** option (SQL name:bLogWorkItem), in the **General** tab of the activity detail must be checked.

Use the Tools/ Tasks in progress menu item to access the list of workflow tasks to be carried out.

Workflow activity assignee

Assignees are appointed to perform tasks resulting from **Question** or **User action** type workflow activities. Assignees are not involved in **Automatic action** or **Test / script** type activities.

Workflow assignees are stored in the table of workflow roles (SQL name: amWfOrgRole). Use the Tools/ Workflow/Roles menu item to access the table of workflow roles.

Definition of a workflow execution group

Workflow execution groups enable you to classify the workflow schemes that you define. The execution group to which a given workflow scheme belongs is indicated in the **Execution group** field (SQL name: GroupName) in the **General** tab of the workflow detail.

General overview

The first step in implementing workflow under AssetCenter is to define workflow schemes using the graphical editor via the **Tools/ Workflow/ Workflow schemes** menu item. Workflow schemes define activities, events and transitions. They can reference AssetCenter actions and employees (workflow assignees).

Workflow schemes are interpreted by workflow engines. The workflow engines in question are run by either AssetCenter Server, or agents of AssetCenter.

In reaction to events, workflow engines trigger and monitor workflow instances:

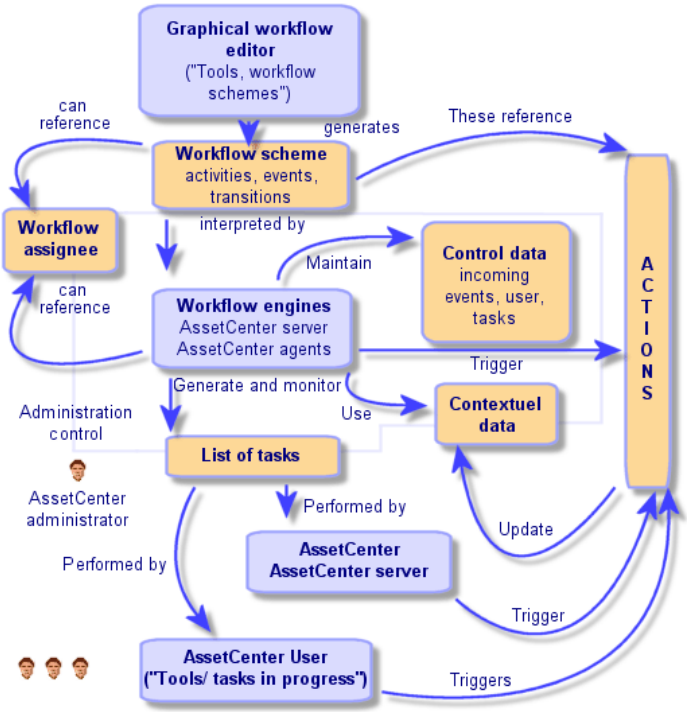
- The workflow engines generate the tasks to be performed.
- They monitor these tasks and events leading to activities.

- They also keep a record of the course of events, by logging incoming events and user tasks to be performed.

Workflow tasks are either performed by the workflow engines or by AssetCenter users. Once they are carried out, they activate events that are then taken into account by the workflow engines.

The following diagram gives an overview of how workflow is implemented in AssetCenter:

Figure 8.2. Workflow in AssetCenter - overview

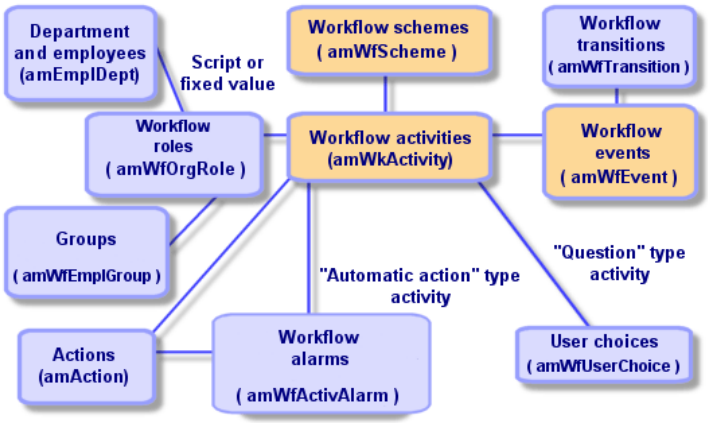


The processing of workflow instances varies according to the nature of activities and events and the way in which the workflow engines controlling them have been configured.

Main tables involved in workflow

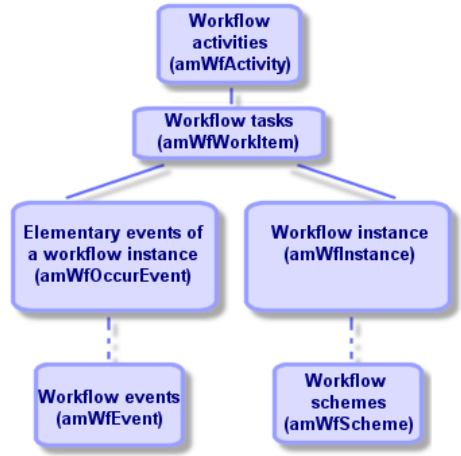
The following diagram presents the main tables involved in workflow and the links between them. The tables are identified by their label and SQL name:

Figure 8.3. Workflow - main tables allowing you to define a workflow scheme



The following diagram presents the main tables involved when a workflow instance is in progress:

Figure 8.4. Workflow - main tables involved in a workflow instance being executed



Using the graphical workflow editor

Use the Tools/ Workflow/ Workflow schemes menu item to access the workflow schemes. This menu item is reserved for the AssetCenter administrator. Use the Tools/ Workflow/ Workflow schemes menu item to access the workflow schemes. This menu item is reserved for AssetCenter administrators.

The **Activities** tab in the detail of workflow scheme includes a graphical editor that enables user-friendly creation of the workflow schemes.

This section details how you use this graphical editor in order to create, modify or delete workflow schemes:

- Activities
- Events
- Transitions
- Other functionality

Activities

To create an activity:

- Either right-click an empty area of the **Activities** tab, then select the **Add an activity** command from the shortcut menu. The detail of the activity is displayed.
- Or alternatively, click the button, then click in the graphical zone. The detail of the activity is displayed.

To delete an activity:

- Either select the activity by clicking it then press the "Delete" key.
- Or select the activity and select the **Delete** command from the shortcut menu.
- Or select the activity and click the button.

To modify the detail of an activity:

- Either right-click the activity and select the **Detail of activity** item from the shortcut menu.
- Or double-click the activity.

Events

Database or Periodical type event

To add an output event to an activity:

- Either right-click the activity, then select the **Add event** command from the shortcut menu.
- Or select the activity, then click the button.

To delete a **Database** or **Periodical** type output event from an activity:

- Either select the event, then press the "Delete" key.
- Or double-click the event and select the **Delete** command from the shortcut menu.
- Or select the event and click the button.

To modify the detail of a **Database** or **Periodical** type event:

- Either double-click the event.
- Or select the event then select the **Detail of event** item from the shortcut menu.

System event

You can create and delete **System** events from the activities at their origin.

To modify a **System** event, use one of the following methods as appropriate:

- To modify the processing method of an event (**Processing** field (SQL name: seProcessingMode) in the detail of the event), proceed as if modifying a **Database** or **Periodical** type event.
- Else, edit and modify the detail of the activity at the origin of the event.

Transitions

To create a transition:

1 Click the starting event to select it.

2 Hold the mouse button down and drag to the target activity.

To delete a transition:

- Either click the transition to select it, then press the "Delete" key.
- Or select the transition and select the **Delete** command from the shortcut menu.
- Or select the transition and click the  button.

To modify the detail of a transition:

1 Click the transition to select it.

2 Select the **Detail of transition** item from the shortcut menu.

To modify the source and/ or target of a transition:

1 Select the transition.

2 Drag the end you want to modify.

Other functionality

The graphical editor also allows you to:

- Drag and drop linked activities and transitions.
- Enlarge or reduce the scheme using the **Zoom** slider or the  button.

How to implement workflow

Workflow under AssetCenter makes it possible to automate the procedures of your company. Here are the steps to follow:

1. Analyze the procedures of your company that you want to formalize.
2. Create:
   1. Workflow roles.
   2. Actions.
3. Create workflow schemes for which you define:
   1. Activities, events and transitions.
   2. Alarms.
4. Define the appropriate workflow execution groups. Associate each workflow scheme with a workflow execution group.
5. Launch AssetCenter Server on one or more machines. For each instance of AssetCenter Server, define the workflow execution groups to be monitored and the monitoring parameters.

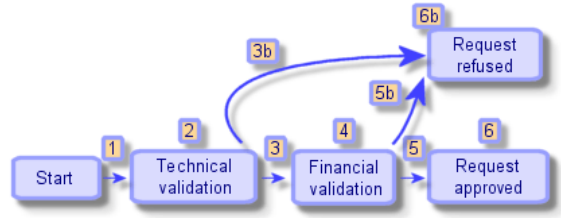
Example of workflow used in request approval

This section details a simple example of workflow.

Aim

The aim of this workflow scheme is to automate the purchase request process according to the following:

Figure 8.5. Workflow - request validation



The steps of the workflow scheme are as follows:

1 The workflow instance starts as soon a purchase request is to be validated, i.e. when the **Req. status** field (SQL name: seStatus) in the purchase request detail indicates **Awaiting approval**.

2 The request first undergoes technical validation. This step consists of submitting the request to the departmental supervisor for approval. They are informed by a message. A reminder alarm is programmed to be triggered if the request is not dealt with by the approver before the end of the next business day following the issuing of the approval request message.

3 If the person responsible validates the purchase request, the next step is financial validation.

   Otherwise the request is refused (3b).

4 Financial validation consists of submitting the request to the company's financial controller, Mr. Gerald Colombo. They are also notified by mail and a approval reminder alarm is programmed.

5 If the financial controller validates the purchase request, the purchase request is approved.

   Otherwise the request is refused (5b).

6 If the purchase request is approved, AssetCenter sets the **Req. status** field in the purchase request detail to **Validated**.

   When the purchase request is refused, AssetCenter sets the **Req. status** field in the purchase request detail to **Refused**.

Prerequisites

You must connect to the database under the login **Admin** and configure the messaging system (➔ Messaging [page 111]).

Creating workflow assignees

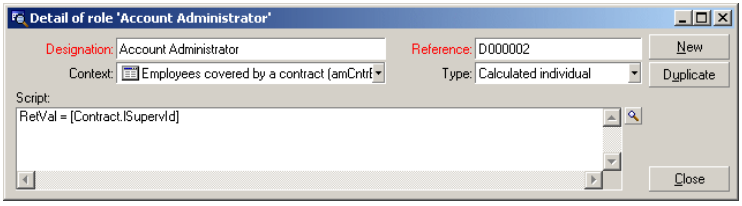
The assignees involved in this workflow scheme are:

- The departmental supervisor of the requester.
- The financial controller of the company, Mr. Gerald Colombo.

The workflow assignees are stored in the table of workflow roles (SQL name: amWfOrgRole). To create them, select the **Tools/ Workflow/Roles** menu item, then click **New**.

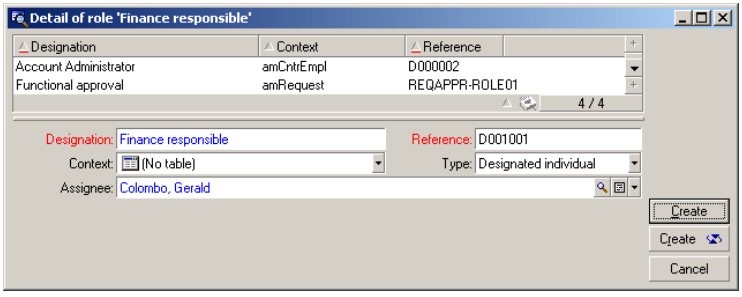
Departmental supervisor of requester

This person is calculated by a script. To define this, populate the detail screen as follows:



Financial controller

This person is designated as Mr. Gerald Colombo. To define him as an assignee, populate the detail screen as follows:

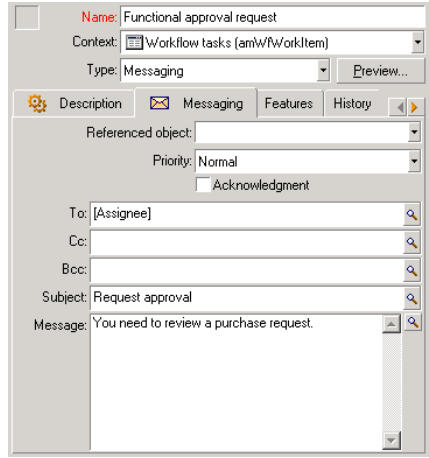


Creating actions

The workflow scheme calls on numerous actions. To create them, select the **Tools/ Actions/ Edit** menu item.

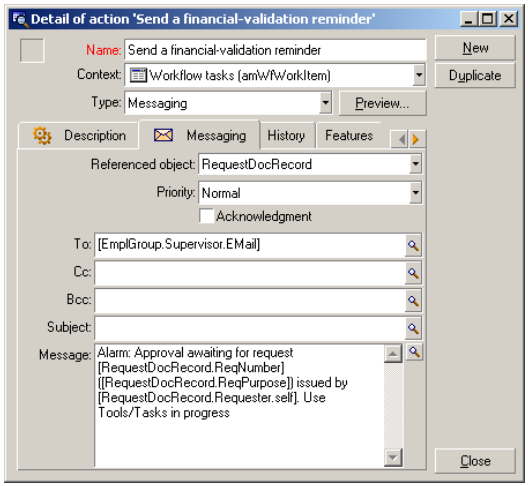
Request for technical validation sent to departmental supervisor of requester

This action is used in the technical validation phase. It makes it possible to notify the person in charge of technical validation of the need to review this request:



Request for financial validation sent to financial controller

This action is used in the financial approval process. It sends a message to the person in charge of financial validation of the need to review this request:



Validation of the purchase request

This action is used at the level of the **Request approved** activity, which will be created later on.

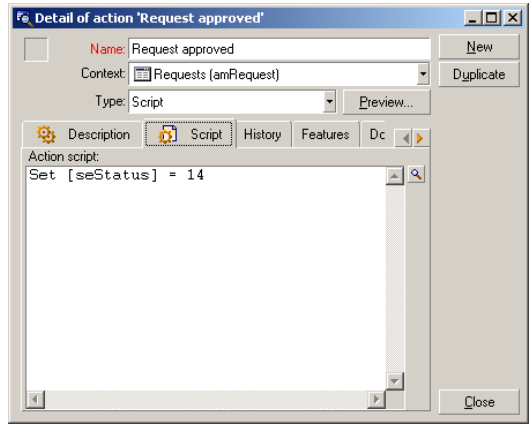
It sets the **Req. status** field (SQL name: seStatus) in the purchase request detail to **Validated**. This action is a **Script** type action.

The **Req. status** field is a system itemized list. To see the list of its values, display the help on this field:

1 Right-click the **Req. status** field in the detail of the request.

2 Select the **Help on this field** menu item from the shortcut menu: The value displayed as **Validated** is stored in the database as **14**.

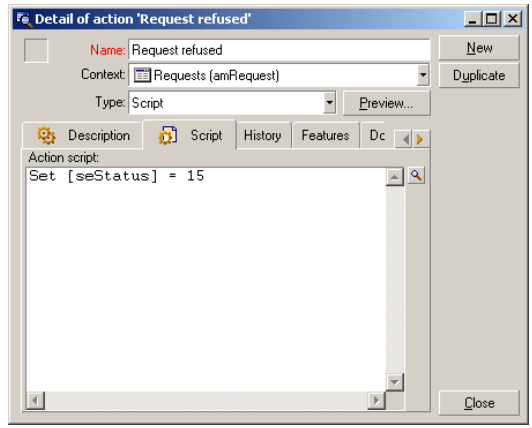
The action is as follows:



Refusal of the purchase request

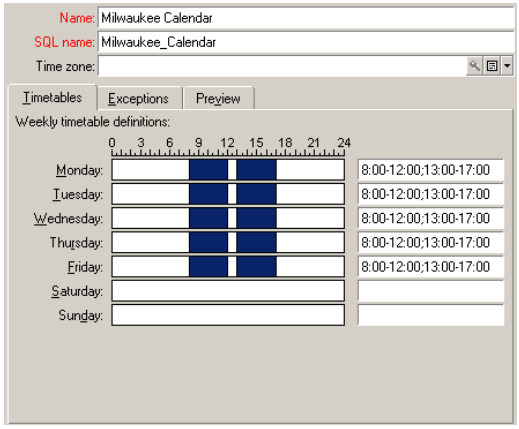
This action is used at the level of the **Request refused** activity, which will be created later on.

It is similar to the **Request approved** action, but the **Req. status** field (SQL name: seStatus) in the **General** tab of the request detail must be set to **Refused**.



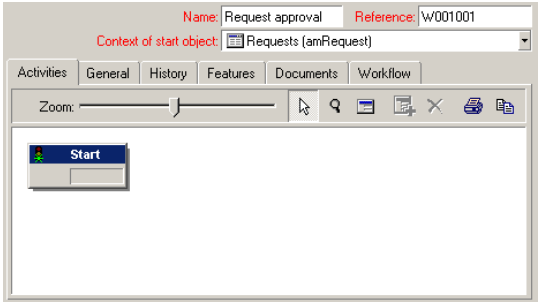
Creating the calendar

Use the **Tools/Calendars** menu item to access the list of calendars. This calendar is associated with workflow scheme activities. It enables you to set deadlines for instances of the workflow scheme:



Preparing the workflow scheme

1. Launch the **Tools/ Workflow/ Workflow schemes** menu item.
2. Click **New**.
3. Assign the name **Request approval** to the workflow.
4. Indicate the context of the start object that will apply by default to all activities in the workflow scheme. In our case, this is the table of requests (SQL name: amRequest).
5. Click **Create**: The start activity (**Start**) is automatically created by AssetCenter in the graphical editor in the **Activities** tab.



Creating activities

Activities can be created graphically in the **Activities** tab of the workflow detail:

1 Click outside of a workflow object.

2 Right-click.

3 Select the **Add activity** menu item from the shortcut menu: The detail of the activity is displayed.

Creating the Technical validation activity

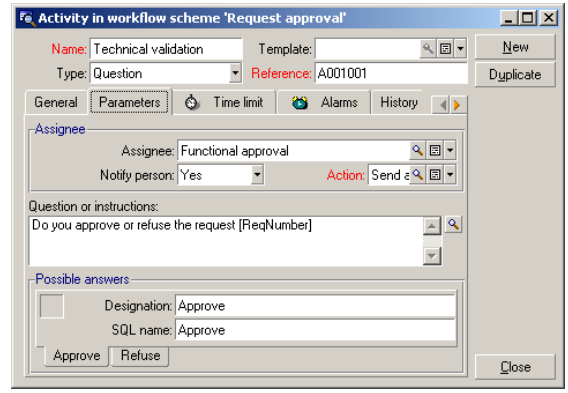
1 Assign a name to the **Technical validation** activity.

2 Since the activity submitting the request to the departmental supervisor for approval, select the value **Question** from the **Type** itemized list (SQL name: seType).

3 The **Context** field (SQL name: ContextTable) in the **General** tab is not modified.

Configuring the Technical validation activity

1 Populate the **Parameters** tab as shown below:



2 Specify the question to be asked:

1 The text of the question references the number of the purchase request.

2 There are two possible responses: Refuse or approve. To add a sub-tab describing a response to the question, right-click the label zone of the sub-tabs and select **Add linked record** or **Duplicate link**.

3 Indicate to whom the question is addressed in the **Assigned** field (SQL name: Assignee). In our case, the assignee is the departmental manager of the requester. This person was created in the table of workflow roles in the preliminary steps.

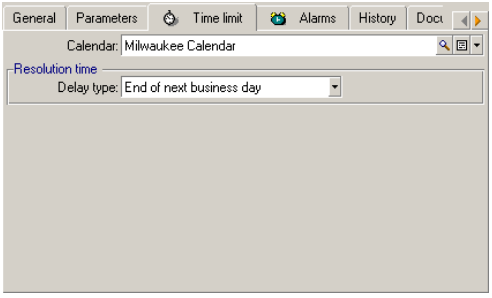
4 In order for the assignee to be notified of the need to review this request:
   1 Set the **Notify person** field (SQL name: bNotifAssignee) to **Yes**.
   2 Specify the action to be carried out: This is the **Request for technical approval** action created in the intermediary step. This action is automatically triggered when a purchase request is submitted to technical approval.

Defining the time limit for performing the Technical validation activity

In the **Time limit** tab of the activity detail:

1 Specify the calendar of business days attached to the activity. This calendar is taken into account when calculating time limits. Select the **Milwaukee calendar** created in the preliminary step.

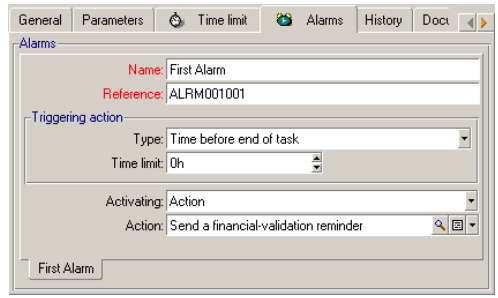
2 Define the time after which the decision must be taken in relation to the time at which the activity is triggered. In our case, the assignee must respond to the question before end of the next business day.



Defining an alarm for the Technical validation activity

In the **Alarms** tab of the activity detail, define a reminder alarm in case the decision is not taken before the end of the time limit specified in the **Time limit** tab.

In order to simplify things, the alarm will trigger the **Request for technical approval** action:



It is possible to define further alarms using the **Add linked record** command from the shortcut menu.

Events

Once the activity is created, AssetCenter creates two system events **Approve** and **Refuse** corresponding to the two possible responses to the question:

When these events occur, an AssetCenter agent logs them in the table of workflow elementary events (SQL name amWfOccurEvent).

The following activities are triggered either by AssetCenter Server directly or by AssetCenter Server according to the configuration of the workflow events:

- If the **Processing** field (SQL name: seProcessingMode) is set to **Log event and process immediately** or **Process event immediately without logging**, AssetCenter triggers the following action.
- If the **Processing** field (SQL name: seProcessingMode) is set to **Log event and process by server**), AssetCenter Server triggers the following action.

  By default, the (**Processing** field (SQL name: seProcessingMode) in the **General** tab of event detail is set to **Log event and process by server**.

Creating the Financial validation activity

This activity is similar to the previous one.

The assignee differs: In this case, it is the financial controller of the company, Mr. Gerald Colombo (a designated individual). He was created in the table of workflow roles in the previous step. In order to notify him, select the **Request for financial validation** action.

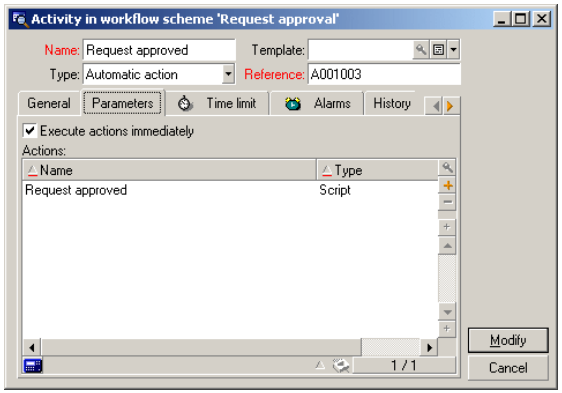
Creating the Request approved activity

When the request succeeds in passing the two validation steps, it is approved.

The **Request approved** activity is one of the possible endings of the workflow scheme.

This activity needs to modify the detail of the request in order to show the request as being approved.

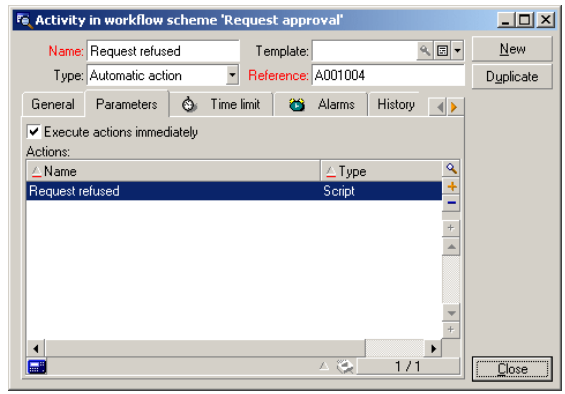
This activity is thus an **Automatic action** type activity: The action to be executed is the **Request approved** action created in the preliminary step.



Creating the Request refused activity

The **Request refused** activity is similar to the **Request approved** activity.

In this case, the detail of the request needs to be modified in order to indicate that the request has been refused. The action to be executed is the **Request refused** action created in the preliminary step.



Configuring events created at the same time as activities

When the activities were created during the previous steps, the following events were also created:

- **Functional approval** activity:
  - **Approve** event
  - **Refuse** event
- **Financial validation** activity:
  - **Approve** event
  - **Refuse** event
- **Request approved** activity
  - **Executed** event
- **Request refused** activity:
  - **Executed** event

If you select each event one after the other and perform the following options, then you won't have to rely on the AssetCenter Server:

1 Right-click on the event name.

2 Select the **Detail of event** menu.

3 Select the **General** tab.

4 Select the **Log event and process immediately** value to save the **Processing** field (SQL name: seProcessingMode).

5 Click **Modify**.

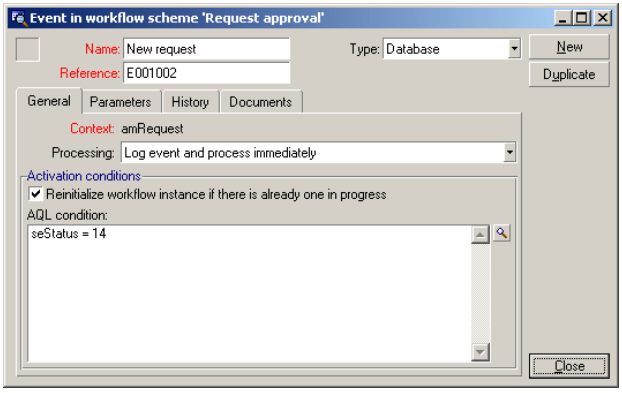
Creating the start event

Events that trigger a workflow instance are associated with the **Start** activity.

To define a **Start** event, right-click the empty event zone in the **Start** activity and select **Detail of event**.

1 In our case, the workflow instance is triggered when the **Req. status** field (SQL name: seStatus) in the detail of a request detail is set to **Awaiting approval**.

The start event is therefore a **Database** type event and its parameters are described in the **General** tab as shown in the screen below:



2 Set the **Processing** field (SQL name: seProcessingMode) in the **General** tab of the event to **Log event and process immediately**.

3 In the **Parameters** tab of the event:

1 Check the **Update** box (SQL name: bUpdate).

2 Specify the **seStatus** field in the **Fields monitored** field (SQL name: MonitFields).

Creating transitions

Once the activities are created, they now need to be linked by transitions.

To create a transition:

1 Click the start event of the transition.

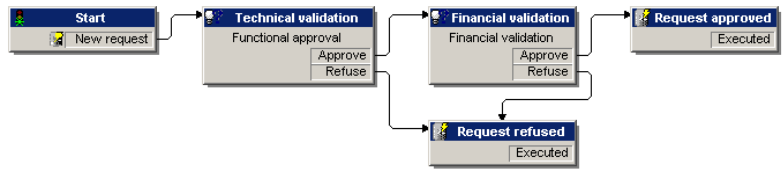
2 Hold the button down and drag to the target activity.

In this case, the transitions to be created are the following:

- From the start event to the **Technical validation** activity.
- From the **To approve** event of the **Technical validation** activity to the **Financial approval** activity.
- From the **To approve** event of the **Financial validation** activity to the **Request approved** activity.
- From the **Refuse** events of the **Technical validation** and **Financial validation** activities to the **Request refused** activity.

We obtain the following workflow scheme:

Figure 8.6. Workflow - request validation scheme



Example of activating a workflow instance

We now need to verify that the workflow scheme functions correctly.

In order to do this, we need to do the following:

1 Populating the Departments and employees table [page 137]

2 Creating a purchase request to be approved [page 138]

3 Controlling the instance [page 139]

Populating the Departments and employees table

Before creating the purchase request to be approved, it is important to define the requester and the corresponding departmental supervisor in the table of

departments and employees. The supervisor needs to have the appropriate rights in order to perform the following operations:

1 Create the requester **Jerome Carpenter**, belonging to the **IS department**.
2 Attribute a **Login**, password and user profile allowing him to enter a purchase request (**Profile** tab in the detail of the corresponding record).
3 Assign the value **AM:Carpenter** to the **Email** field (SQL name: EMail).
4 The supervisor of the **IS department** is **Philip Chavez**.
5 To simplify the following operations, grant administrative rights for the database to Philip Chavez: Display the **Profile** tab in the detail of the corresponding record and check the **Administration rights** box (SQL name: bAdminRight). Specify the **Login** (SQL name: UserLogin) and the password of Philip Chavez.
6 Assign the value **AM:Carpenter** to the **Email** field (SQL name: EMail).
7 Select the employee **Gerald Colombo**.
8 Assign the value **AM:Colombo** to the **EMail** field (SQL name: EMail) and **Colombo** to the **Login** field (SQL name: UserLogin).

Creating a purchase request to be approved

The following step consists of creating a purchase request to be approved:

1 Connect to the demonstration database using the login name of Jerome Carpenter.
2 Launch the **Procurement/ Purchase requests** menu item under AssetCenter.
3 Click **New**.
4 Select the **Create a new custom request** option.
5 Click **OK**.
6 In the **Requester** field (SQL name: Requester) in the **General** tab of the detail of the request, select the record **Carpenter**.
7 Set the **Req. status** field (SQL name: serStatus) in the detail of the request to **In preparation**.
8 Confirm the purchase request: The start event occurs and an AssetCenter logs the event in the table of workflow elementary events (SQL name: amWfOccurEvent).

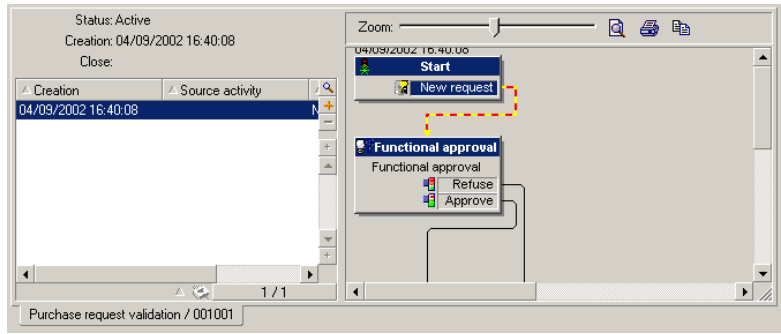
Controlling the instance

In order to verify that the workflow instance is functioning correctly, launch AssetCenter and connect to the demonstration database using the login of Philip Chavez, the departmental supervisor of "Jerome Carpenter".

Viewing the workflow instance

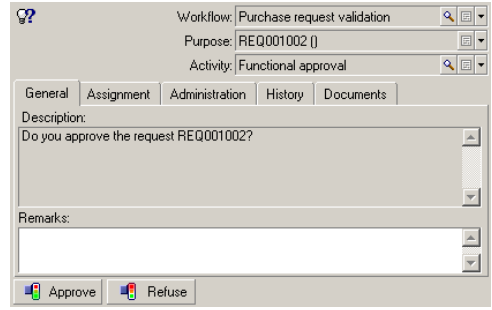
Display the detail of the purchase request that you have created: The **Workflow** tab lists the current workflow instances for the request. Each instance is described in an individual sub-tab.

- The left part of the sub-tab list the events that have occurred.
- The right part shows the status of the instance. In our case, the **Technical validation** task should be flashing.



Viewing the task to be carried out

1 Select the **Tools/ Tasks in progress** menu item: The departmental supervisor "Philip Chavez" can thus view the detail of the assigned task:



The **General** tab displays the question that you have defined in the **Question** tab of the **Technical validation** activity.

The **Assignment** tab describes who is in charge of the task and the corresponding deadline. This date is calculated using the information contained in the **Time limit** tab in the detail of the **Technical validation** activity, and the date of creation of the task (i.e. the date of activation of the transition by AssetCenter Server).

You can click **Detail** to access the detail of the request giving rise to the task.

2 Simply click either the **Approve** or **Refuse** buttons to carry out the task. It is also possible to enter a comment concerning the decision in the **General** tab.

Context of a workflow

Each workflow instance has its own specific context.

Defining the context of a workflow instance

When defining a workflow scheme, you define:

- A default context at the top of the workflow detail.
- A context for all events, transitions and workflow activities (in the detail screen of a transition or in the **General** tab of the detail of events and activities). This context is linked to the default context of the workflow instance.

In both cases, the context is a table.

Object referenced by a workflow instance

When a record fulfills the activation conditions defined in a workflow scheme, a workflow instance is triggered. The record constitutes the object referenced at start event level.

When the workflow instance is running, the referenced object can change according to the context defined at activity, event and workflow transition level.

Example: A workflow instance is triggered when a purchase request is approved. It creates a purchase order according to this request. If request R1 is approved, it constitutes the referenced object of the start event. The referenced object then becomes purchase order PO1, i.e. the order generated from the purchase request.

Limiting workflow instances in progress for a given object

One single active workflow instance for an object option (SQL name: bUniqueActive)

AssetCenter makes it possible for you to limit the number of concurrent workflow instances for a given object using the **One single active workflow instance for an object** option in the **General** tab of a workflow detail.

If an output event of the "Start" activity giving rise to a second workflow instance for an object occurs, the **One single active workflow instance for an object** and **Reinitialize workflow instance if there is already one in progress** options (SQL name: bReinitialize) (**General** tab of the detail of the event), determine the outcome:

The following table resumes the different possible cases:

		One single active workflow instance for an object option in the General tab of the workflow scheme.	
		Validated	**Not validated**
Reinitialize workflow instance if there is already one in progress option in the General tab of the output event of the Start activity.	**Validated**	If there is already a workflow instance in progress for the object, it is stopped and a new workflow instance started.	
	Not validated	If there is already a workflow instance in progress for the object, the event is ignored (no new workflow instance).	A new workflow instance is created.

Example of application:

In the case of a workflow scheme intended to deal with purchase request approvals, it may be useful to:

- Check the **One single active workflow instance for an object** option, in order for a given purchase request to be subject to one single approval process.
- Check the **Reinitialize workflow instance if there is already one in progress** option at the level of the start event in order to restart the instance if the composition of the purchase request is modified.

Workflow roles

Tasks resulting from certain workflow activities must be carried out by an assignee.

Note:

Activity assignees are only involved in **Question** or **User action** type activities. **Automatic action** or **Test / script** type activities do not have assignees.

Activity assignees are selected in the table of workflow roles (SQL name: amWfOrgRole). Use the Tools/ Workflow/Roles menu item to access the table of workflow roles.

Workflow role type

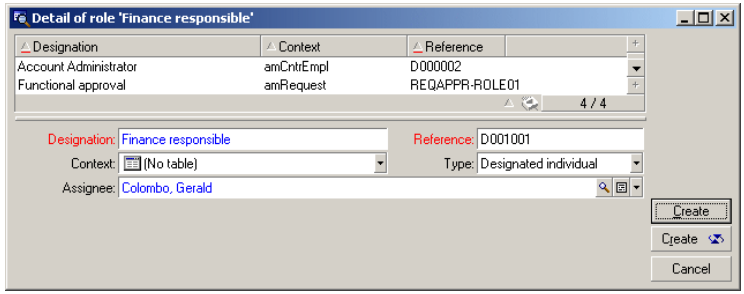
The are several possible types of workflow roles (**Type** field (SQL name: seType) in the workflow role detail):

- **Designated individual**.
- **Calculated individual**.
- **Group**.

Designated individual

In this case, the assignee is selected in the table of departments and employees directly.

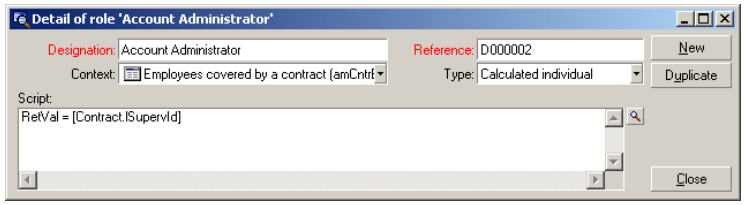
Example:



Calculated individual

In this case, the assignee belongs to the table of departments and employees but is calculated by a script.

Example:



Group

In this case, the assignee is selected in the table of employee groups (SQL name: "amEmplGroup").

Defining the assignee of an activity

The **Assigned** field (SQL name: Assignee) in the **Question** tab (**Question** type activity) or the **Action** tab (**User action** type activity) lets you define the assignee for a **Question** or **User action** type activity.

Workflow activities

Activities can be divided into two groups:

- Those requiring the interaction of a user: **Question** and **User action** type activities (**Type** field (SQL name: seType) at the top of an activity detail).
- Those that are performed automatically: **Automatic action** and **Test / script** type activities.

The value of the **Type** field of an activity determines which tabs are displayed in the activity detail.

This section describes the following activities:

- Question type activities.
- User-action type activity.
- Automatic-action type activity.
- Test/ Script type activities.
- Start activity
- Activity templates
- Triggering activities

Question type activities

These activities require the involvement of a user, specified in the **Assigned** field (SQL name: Assignee).

Question type activities are defined by:

- A question or instructions.
- Possible responses.

Examples:

- In the course of a purchase approval process, a request issued by an employee is submitted to the departmental supervisor.
- A **Question** type activity can also be used as a checkpoint to confirm that a task has been carried out. In this case, there would be, for example, one single possible response.

Set the **Type** field (SQL name: seType) to **Question**. The **Question** tab will be displayed.

Specify:

1 The record in the table of workflow roles corresponding to the assignee. This assignee can be notified via an AssetCenter action. In order to do this, you just need to populate the **Notify person** field (SQL name: bNotifAssignee) in the **Question** tab appropriately.

Note:

The action notifying the assignee is triggered as soon as the task to be performed is created; i.e. as soon as the transition triggering the activity is activated.

The assignee uses the **Tools/ Tasks in progress** menu item to access the detail of the tasks to be carried out.

2 The text of the question or the instructions to follow.

3 The possible responses. Each response is described in a sub-tab. It is identified by its description and its SQL name. To add, duplicate or delete a response, right-click the sub-tab label zone and select **Add linked record**, **Duplicate linked record** or **Delete link** from the shortcut menu.

Note:

Each response automatically creates an output event for the activity.

User-action type activity

These activities require the involvement of a user, called the "assignee". The assignee is shown in the **Assigned** field (SQL name: Assignee).

Their definitions include:

- Instructions to follow.
- A wizard to execute.

Set the **Type** field to **User action** in order for the **Action** tab to be displayed.

Specify:

- The instructions to be followed.
- The wizard to be executed.
- The record in the table of workflow roles corresponding to the assignee. This assignee can be notified via an AssetCenter action. In order to do this, you just need to populate the **Notify person** field (SQL name: bNotifAssignee) in the **Action** tab appropriately.

Note:

The action notifying the assignee is triggered as soon as the task to be performed is created; i.e. as soon as the transition triggering the activity is activated.

The assignee uses the **Tools/ Tasks in progress** menu item to access the detail of the tasks to be carried out.

Note:

An **executed** event is automatically created as an output event of the activity.

Example: When managing deliveries, a wizard can be used to help the user take delivery in full or in part of purchase order lines awaiting delivery.

Automatic-action type activity

These activities are carried out automatically by AssetCenter or AssetCenter Server.

Description

Automatic action type activities list actions to be executed.

Example: In an "Asset move" operation, an **Automatic action** type activity automatically modifies the location of all assets whose parent assets have been moved.

Set the **Type** field to **Automatic action** in order for the **Actions** tab to be displayed.

Indicate the list of actions to be executed here.

Note:

An **executed** event is automatically created as an output event of the activity.

Execution

The workflow engine that activates the transition triggering the activity immediately executes the activity's actions. Depending on the selected options,

these actions will be processed by AssetCenter Server or an agent in AssetCenter.

- If you select the **Execute actions immediately** option (SQL name: bExecImmediately), the workflow engine that activates the transition triggering the activity automatically executes the actions of the activity.
- Otherwise, the tasks are carried out by AssetCenter Server during its next verification cycle.

Test/ Script type activities

These activities are automatically performed by AssetCenter or AssetCenter Server.

Description

They are defined by a script and possible results.

Example: In the area of stock and purchase request management, a test/ script type activity can be used to verify that items referenced in purchase order lines are available in stock and not reserved. If this is the case, the activity can trigger a **Question**-type activity that asks the requesters if they want to reserve the item in stock.

Set the **Type** to **Test / script** in order for the **Test** tab to be displayed.

Specify:

- The test script to be executed.
- Possible results. Each result is described in an individual sub-tab. It is identified by its description and SQL name. To add, duplicate or delete a possible result, right-click the sub-tab label zone and select **Add linked record**, **Duplicate linked record** or **Delete link** from the shortcut menu.

Note:

Attention: The SQL names of each result must correspond to the return values of the test script.

Note:

Each result automatically creates an output event for the activity.

Execution

The workflow engine that activates the transition triggering the activity immediately executes the activity's actions. Depending on the selected options, these actions will be processed by AssetCenter Server or an agent in AssetCenter.

- If you check the" Execute actions immediately option" (SQL name: bExecImmediately), the workflow engine that activates the transition triggering the activity automatically executes the actions of the activity: According to the mode of processing that you have selected for the event that triggers the transition, either AssetCenter Server or an agent of AssetCenter executes the actions.
- Otherwise, the tasks are carried out by AssetCenter Server during its next verification cycle.

Start activity

The **Start** activity is the starting point of a workflow scheme.

It is mandatory and created automatically when you create a workflow scheme. It is not possible to edit the detail.

It does not define the work to be performed.

The output events of the **Start** activity trigger the workflow instance.

Activity templates

Activity templates facilitate the creation of workflow scheme activities.

They are stored in the table of activities (SQL name: "amWfActivity").

Use the Tools/ Workflow/Activity templates menu item to access the list of activity templates.

 Note:

Warning: In order for the information contained in the detail of an activity template (activity type, etc.) to be automatically copied over to the level of the activities referencing this template (**Template** field (SQL name: Template) in the activity detail), the appropriate default values for the fields and links in the detail of an activity need to be defined by an AssetCenter administrator.

Triggering activities

In order for an activity to be triggered, the **Input condition** field (SQL name: seInCond) in the **General** tab of the activity detail must be populated. This condition concerns the transitions that trigger the activity.

- If there is only one transition that can trigger the activity, the transition just needs to be activated by (by AssetCenter or AssetCenter Server) in order for the activity to be triggered.
- If there are several transitions that can trigger the activity:
  - If the input condition of the activity is **AND**, all the transitions must be activated in order for the activity to be triggered.
  - If the input condition of the activity is **OR**, only one of the transitions needs to be activated in order for the activity to be triggered.

Note:

If the input conditions of an activity are complex (combinations of **AND** and **OR**), you can create a sequence of intermediary **Test / script** type activities to achieve this.

Tasks

This section explains how workflow tasks are created and executed:

Creating tasks

When a transition triggering an activity is activated, a task to be carried out is automatically created by the workflow engine that activated the transition.

According to the option selected in the **Log task** field (SQL name: bLogWorkItem) in the **General** tab of an activity, this task is logged to the table of workflow tasks (SQL name: WkElem).

The **Log task** option is automatically validated.

- For **Question** or **User action** type activities.
- For **Automatic action** or **Test / script** type activities, for which the **Execute actions immediately** option (SQL name: bExecImmediately) is not selected.

Warning:

If a task is not logged, it is not possible to create workflow alarms associated with this task: The **Time limit** and **Alarms** tabs in the detail of an activity are not displayed if the **Log task** option is not selected.

The task is performed differently according to whether user involvement is required or not.

Automatic action or test/ script type activity

If the task results from an **Automatic action** or **Test / script** type activity whose **Execute actions immediately** option (SQL name: bExecImmediately) is selected, the task is immediately executed by the workflow engine that activated the transition which created the task. This can be either AssetCenter Server, or an AssetCenter agent.

Otherwise, AssetCenter Server verifies at regular intervals whether is needs to execute workflow task and executes them if necessary.

The frequency with which AssetCenter Server monitors workflow functions is defined in the options of AssetCenter Server.

Displaying the list of tasks in progress

The Tools/ Tasks in progress menu item allows you to display the list of tasks that need to be carried out.

The list displayed depends on the person connected to the database:

- An AssetCenter administrator can see all tasks in progress for all workflow instances.
- A workflow assignee sees:
  - The tasks they have to perform.
  - The tasks that are assigned to groups to which they belong but that are not assigned to a given person.

An administrator can also access the list of tasks in progress for a given activity from a workflow scheme detail. To do this:

1 Right-click an activity.

2 Select **Tasks in progress** from the shortcut menu.

Note:

The displayed list is only a restricted view of the table of workflow tasks (SQL name: "amWfWorkItem": It shows the tasks that are to be carried out.

Performing a user task

A workflow assignee can access the list of tasks to be performed via the Tools/ Tasks in progress menu item.

Warning:

If the user connected to the database is the AssetCenter administrator, the **Tools/ Tasks in progress** menu item displays all tasks to be carried out. Otherwise if the connected user is not the administrator, the **Tools/ Tasks in progress** menu item only displays those tasks that are assigned to them and those tasks assigned to groups to which they belong.

To access the detail of the object referenced by the task, click **Detail**.

To perform the task to be carried out, display the **General** tab of the task:

- If the activity resulting from the task is a **Question** type task, the **General** tab displays the text of the question or the instructions to be followed. The possible results have corresponding buttons. Click the appropriate button. If needed, you can also enter a comment concerning the decision taken.
- If the activity is a **User action** type activity, simply click the **Wizard** button to launch the wizard to be performed.

Assigning user tasks

The information concerning the assigning of user tasks appears in the **Assignment** tab in the detail of the task.

If you have the necessary rights, you can modify the assignment of a user task:

- Value of the **Assignment** field (SQL name: seAssignment).
- Assignee of the task.

Administrating a workflow task

The information concerning the administration of a workflow task is contained in the **Administration** tab of the task detail.

This information is available to users with administration rights only.

Events

Events are associated with activities. They trigger transitions to other activities.

There are three possible system types for an event at the level of an activity. The system type of an event is defined by the **System type** field (SQL name: seType) in the detail of the event:

- **System** event.
- **User** event.
- **Alarm** event.

System event

System events are automatically defined by AssetCenter when creating/ modifying activities.

They correspond to the different possible outcomes (results) of the work carried out in an activity:

- Responses to a **Question** type activity.
- Results of a **Test / script** type activity.
- Event **executed** in the case of a **User action** or **Automatic action** type activity.

Example: If an activity asks a question for which the possible answers are "Yes" and "No", two system events are created at activity level, called "Yes" and "No".

Alarm event

Alarm events of an activity are created when you define the activity alarms that trigger events.

Such an alarm is defined in the **Alarms** tab of the activity detail. The event carries the same name as the alarm.

User event

User events are independent of the tasks carried out within an activity. They are created manually via the graphical workflow editor (**Add event** shortcut menu command).

Note:

Events associated with the **Start** activity are user events.

There are two types of **User** events (**Type** field (SQL name: seMonitoringType) at the top of an event detail):

- **Database**
- **Periodical**

Database type event

Database type events allow you to activate workflow instances on specific records.

A **Database** type event occurs:

- When the general activation conditions specified in the **General** tab are fulfilled.
- When certain triggering parameters are verified at the level of the records being monitored.

The parameters that trigger a **Database** type event are described in the **Parameters** tab of the detail of the event. The following information is specified:

- The records to be monitored (These records can be records in the table indicated in the context or linked records.) If the records to be monitored are records linked to the table indicated in the context, specify the corresponding link in the **Link / context** field (SQL name: LinkToMonitTable).
- Activation conditions for the event concerning the records being monitored. To specify the conditions of activation, you can:

- Check one or more appropriate boxes from among the **Insert** (SQL name: bInsert), **Update** (SQL name: bUpdate), and **Delete** (SQL name: bDelete) boxes.

  If you check the **Insert** box, records created are taken into account.

  If you check the **Update** box, you can specify the fields that, when modified, must be taken into account in the **Fields monitored** field (SQL name:MonitFields). To indicate several field names, use commas to separate them. If you leave the field empty, modified fields are not taken into account.

  If you check the **Delete** box, destroyed records are taken into account.

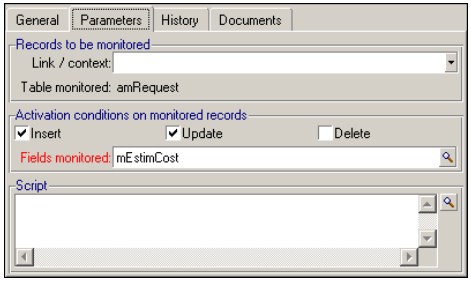
  Warning:

  It is not possible for the activation condition of the event to be the destruction of the object referenced by the context.

- Write a script in the **Script** zone (SQL name: memScript). If you write a script and check one or more of the **Insert**, **Update** and **Delete** boxes, the script restricts the activation conditions.

Example: If an event is to be triggered when the total price of an existing request is modified, populate the **Parameters** tab as follows:

Figure 8.7. Parameters tab of a Database type action



As soon as a **Database** type event occurs, it is taken into account by the AssetCenter client machine on which it occurred. The way in which it is processed depends on the option selected in the **Processing** field (SQL name: seProcessingMode) in the **General** tab of the event detail.

For further information, please refer to the section entitled "Processing of events", in the "Workflow" chapter of this manual.

Periodical type event

Periodical type events concern a selection of records in a given table. They allow you to trigger a workflow instance on a periodical basis for each record of the selection.

Example: Every month, the residual values of assets with a "PC" nature are updated.

A **Periodical** type event occurs if the activation conditions indicated in the **General** tab are fulfilled.

In this case, AssetCenter Server triggers the event.

The frequency with which AssetCenter Server triggers **Periodical** type events is defined in the planner in the **Parameters** tab of the detail of the event.

The way in which the event is processed is described in the section entitled "Processing of events", in the "Workflow" chapter of this manual.

General conditions of activation

The conditions of activation for all types of event can be defined in the **General** tab:

AQL condition (SQL name: AQLCond)

The **AQL condition** field specifies the selection of records involved in the workflow scheme.

Reinitialize workflow instance if there is already one in progress (SQL name: bReinitialize)

Note:

The **Reinitialize workflow instance if there is already one in progress** option only appears in the details of events resulting from the "Start" activity.

The **Reinitialize workflow instance if there is already one in progress** option determines what happens if an output event of the **Start** activity concerns an

database object that is already the subject of an instance of this workflow scheme.

What happens depends not only on this option but also on the **One single active workflow instance for an object** option (SQL name: bUniqueActive) in the **General** tab of the workflow scheme.

The following table resumes the different possible cases:

Table 8.1. Limiting workflow instances in progress for a given object - different possible cases

		One single active workflow instance for an object option in the General tab of the workflow scheme.	
		Validated	**Not validated**
Reinitialize workflow instance if there is already one in progress option in the General tab of the output event of the Start activity.	**Validated**	If there is already a workflow instance in progress for the object, it is stopped and a new workflow instance started.	
	Not validated	If there is already a workflow instance in progress for the object, the event is ignored (no new workflow instance).	A new workflow instance is created.

Processing of events

Once the general activation conditions are fulfilled, the way in which events are processed depends on:

- The event "type" (**Type** field (SQL name: seMonitoringType) at the top of an event detail).
- The option selected in the **Processing** field (SQL name: seProcessingMode) in the **General** tab if the detail of an event.

The following table resumes the different ways in which an event can be processed:

Table 8.2. The different ways in which an event can be processed

	Log event and process by server	Log event and process immediately	Process event immediately without logging
Periodical type event	AssetCenter Server triggers the event if the activation conditions are fulfilled. The frequency of triggering is defined in the **Parameters** tab of the event detail. As soon as it occurs, AssetCenter Server logs the event in the table with SQL name **amWfOccurEvent**. The transition is activated later by AssetCenter Server (the frequency with which AssetCenter Server monitors the transitions to be activated is defined in the options in AssetCenter Server).	AssetCenter Server triggers the event if the activation conditions are fulfilled. The frequency of triggering is defined in the **Parameters** tab of the event detail. As soon as it occurs, AssetCenter Server logs the event in the table with SQL name **amWfOccurEvent**. The transition is activated immediately by AssetCenter Server.	AssetCenter Server triggers the event if the activation conditions are fulfilled. The frequency of triggering is defined in the **Parameters** tab of the event detail. As soon as it occurs, AssetCenter Server does not log the event in the table with SQL name **amWfOccurEvent**. But the transition is activated immediately by AssetCenter Server.

	Log event and process by server	Log event and process immediately	Process event immediately without logging
Database type event or system event triggered by AssetCenter (result of a Question or User action type activity, result of a Automatic action or Test / script type activity executed by AssetCenter)	As soon as the event occurs, it is logged in the table with SQL name **amWfOccurEvent** by the AssetCenter client machine. The transition is activated later by AssetCenter Server (the frequency with which AssetCenter Server monitors the transitions to be activated is defined in the options in AssetCenter Server).	As soon as the event occurs, it is logged in the table with SQL name **amWfOccurEvent** by the AssetCenter client machine. The transition is activated immediately by the AssetCenter client machine.	When the event occurs, it is not logged in the table with SQL name **amWfOccurEvent** but the transition is activated immediately by the AssetCenter client machine.
System event triggered by AssetCenter Server (result of a Automatic action or Test / script type activity executed by AssetCenter Server) or event on the activity alarm	As soon as the event occurs, it is logged in the table with SQL name **amWfOccurEvent** by AssetCenter Server The transition is activated later by AssetCenter Server (the frequency with which AssetCenter Server monitors the transitions to be activated is defined in the options in AssetCenter Server).	As soon as the event occurs, it is logged in the table with SQL name **amWfOccurEvent** by AssetCenter Server. The transition is activated immediately by AssetCenter Server.	When the event occurs, it is not logged in the table with SQL name **amWfOccurEvent** but the transition is activated immediately by AssetCenter Server.

Table 8.3. The different ways in which an event can be processed

	Log event and process by server	Log event and process immediately
Periodical type event	AssetCenter Server triggers the event if the activation conditions are fulfilled. The frequency of triggering is defined in the **Parameters** tab of the event detail. As soon as it occurs, AssetCenter Server logs the event in the table with SQL name **amWfOccurEvent**. The transition is activated later by AssetCenter Server	AssetCenter Server triggers the event if the activation conditions are fulfilled. The frequency of triggering is defined in the **Parameters** tab of the event detail. As soon as it occurs, AssetCenter Server does not log the event in the table with SQL name **amWfOccurEvent**. But the transition is activated immediately by AssetCenter Server.
Database type event or system event triggered by AssetCenter (result of a Question or User action type activity, result of a Automatic action or Test / script type activity executed by AssetCenter)	As soon as the event occurs, it is logged in the table with SQL name **amWfOccurEvent** by the AssetCenter client machine. The transition is activated immediately by the AssetCenter client machine.	When the event occurs, it is not logged in the table with SQL name **amWfOccurEvent** but the transition is activated immediately by the AssetCenter client machine.
System event triggered by AssetCenter Server (result of a Test / script or Automatic action type activity executed by AssetCenter Server) or event on the activity alarm	As soon as the event occurs, it is logged in the table with SQL name **amWfOccurEvent** by AssetCenter Server. The transition is activated immediately by AssetCenter Server.	When the event occurs, it is not logged in the table with SQL name **amWfOccurEvent** but the transition is activated immediately by AssetCenter Server.

Using these different processing modes, it is possible to accurately specify how a workflow instance is run.

According to the selections made at the level of:

- Event types
- Event processing modes
- Activities

You can define both synchronous and asynchronous workflow schemes or combine both approaches.

Application: Implementing a synchronous workflow scheme

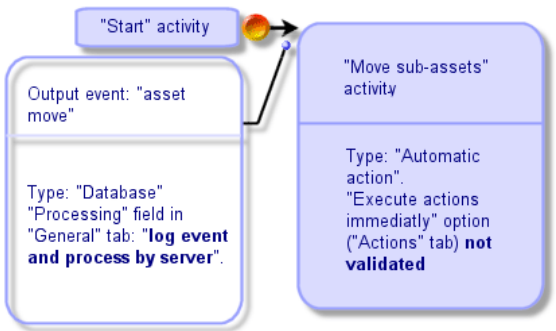
In order to accomplish a synchronous workflow scheme, you need to define:

- **Database** type event that are **Log event and process immediately** (**Processing** field (SQL name: seProcessingMode) in **General** tab of event detail).
- **Automatic action** or **Test / script** type activities, for which the **Execute actions immediately** option (SQL name: bExecImmediately) is selected, and which are triggered by these events.

Example:

Using the workflow scheme described in the diagram below, as soon as an asset changes location, its sub-assets are automatically moved to the same location:

Figure 8.8. Example of synchronous workflow scheme



In this case, when the location of an asset is modified and you click **Modify**:

1. A database transaction starts.
2. The location of the asset is modified.
3. The workflow instance starts up.
4. The workflow transition is activated.

5 The location of the sub-assets is modified.

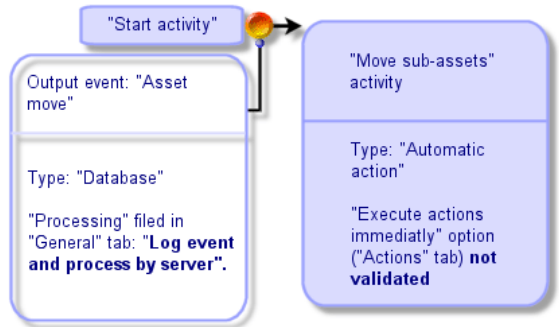
6 Then the full transaction is validated.

If an error occurs during one of the steps, both the locations of the asset and sub-assets are left unmodified.

If the steps are performed successfully, all the locations are modified.

On the other hand, if the same procedure is implemented using an asynchronous workflow scheme as described below, and if an error occurs, the location of the asset can be modified without the location of the sub-assets being modified.

Figure 8.9. Example of asynchronous workflow scheme



Terminal event

Definition

A terminal event ends a workflow instance, even if there are remaining tasks to be performed.

Example:

Figure 8.10. Workflow scheme with terminal event

If a workflow instance occurs as shown above and:

- The output event of activity 1 occurs and triggers activity 2, creating a task to be carried out.
- The terminal event of activity 3 occurs.

Then the workflow instance terminates, even if the task resulting from activity 2 has not been carried out.

Specifying that an event is terminal

When you create a workflow scheme via the graphical editor in the **Activities** tab of a workflow scheme, you indicate that an event is terminal as follows:

1 Right-click the event.

2 Select **Terminal event** from the shortcut menu.

Workflow transitions

Transitions link output events from an activity to other activities.

A event can be linked to several transitions.

If necessary, you can use the **AQL condition** field (SQL name: AQLCond) in the detail of a transition to specify the conditions of activation of the transition.

Workflow alarms and time limits

For each workflow activity, it is possible to define:

- A deadline for performance.
- Alarms linked to this deadline or dates stored in the database. These alarms launch actions.

Warning:

If you do not activate the **Log task** option (SQL name: bLogWorkItem) in the **General** tab of the activity, you cannot define time limits or alarms.

Time limit

The time limit for performance of a workflow activity is defined in the **Time limit** tab of the detail of the activity.

Note:

The **Time limit** tab in the detail of an activity is only displayed if the **Log task** option (SQL name: bLogWorkItem) in the **General** tab of the detail of the activity is validated.

This time limit is defined in relation to the time at which the activity is triggered.

It is associated with a business days calendar.

You can specify a period of time or select one of the three predefined options:

- **End of next business day**
- **End of business week**
- **End of business month**

Warning:

If you specify a period of time, AssetCenter considers that it is expressed in terms of business time and converts it to business hours. Example: If you enter "2 days" this is taken as meaning 48 business hours.

Workflow alarms

It is possible to associate alarms with each activity in the **Alarms** tab of the activity detail.

Note:

The **Alarms** tab in the detail of an activity is only displayed if the **Log task** option (SQL name: bLogWorkItem) in the **General** tab of the detail of the activity is validated.

Time limits

The time limits that trigger alarms can be defined according to:

- A period of time after a date stored in the database (**Time elapsed since start of task** type).
- A period of time before a date stored in the database (**Time before end of task** type).
- A percentage of the time limit for performing the activity (**Time limit** field (SQL name: tsResolDelay) in the **Time limit** tab).

Note:

The periods of time defining workflow time-limits are in terms of business days.

As soon as a task is created, associated workflow alarms are generated.

Workflow time limits are monitored by AssetCenter Server. The frequency of monitoring is defined in the options of AssetCenter Server.

The effect of alarms

Alarms trigger:

- AssetCenter actions.
- Or events. Events triggered by alarms are **Alarm** type events. They carry the name of the alarms that define them.

Workflow execution groups

Workflow execution groups allow you to classify workflow schemes. The execution group to which a workflow scheme belongs is indicated in the **Execution group** field (SQL name: GroupName) in the **General** tab of the workflow detail.

AssetCenter Server monitors the creation of new workflow execution groups.

As soon as AssetCenter Server detects a new workflow execution group G, it creates a new monitoring module **Execution of workflow rules for execution group "G"**.

This mechanism is useful for the following reasons:

- It allows you to define verification timetables for each workflow execution group.
- Different workflow execution groups can be monitored by different instances AssetCenter Server.

Once a workflow execution group is detected, AssetCenter Server monitors and executes the corresponding workflow rules (monitoring alarms, processing of **Periodical** type events, activating transitions, executing tasks, etc).

Workflow tracking

When a table in AssetCenter is defined as the context of the start object of a workflow scheme, a **Workflow** tab is displayed in the detail view of this table.

This **Workflow** tab shows the status of the workflow instances in progress that use this record as the start object.

Each workflow instance is described in a sub-tab that describes the progress of the instance:

- The left part of the sub-tab list the events that have occurred.
- The right part of the sub-tab displays the workflow scheme. The activities to be carried out are shown as flashing. The following steps are grayed out.

Deleting instances of finished workflows

Why should you delete an instance of a finished workflow?

Executing workflow schemes creates workflow instances (in the **Workflow instances** table (amWfInstance)).

These workflow instances are not automatically destroyed, even if they have finished executing (**Status** field (seStatus)).

If you use a large number of workflow schemes, you will obviously have a large number of workflow instances.

This unnecessarily increases the size of the database and can tax AssetCenter's performances.

We thus recommend that you regularly delete workflow instances that have finished executing.

Automating the deletion of terminated workflow instances

To automate the deletion of terminated workflow instances from your working database:

1. Add a field to the **Workflow schemes** table (amWfScheme) to define a deadline for when obsolete workflow instances must be deleted.
2. Create an action that deletes these obsolete workflow instances.
3. Create a workflow scheme that automates the execution of this action.
4. Configure AssetCenter Server to automate the execution of the workflow scheme.

Adding a field to the Workflow schemes table

Add the following field to the **Workflow schemes** table (amWfScheme):

Parameter	Value
SQL name	AutoCleaningDelay
Label	Instance deletion delay
Description	A deadline that, when reached, mandates the deletion of the obsolete workflow instances.
Type	Duration (time span)
Create an index for this field	Do not select this option
Description (**Help** tab)	<LI>Determines how much time should pass before obsolete workflow instances can be deleted.
Example	<LI>-1: Workflow instances are never deleted.<LI>Positive or null value: Workflow instances can be deleted after the deadline has passed.

To learn how to add a field to an existing table, refer to the **Administration** guide, chapter **Customizing the database**, section, **Creating new objects**, sub-section **Creating a field, link or index**.

Populate the Instance deletion delay field

Populate the **Instance deletion delay** field (AutoCleaningDelay) in each workflow scheme you use.

Connecting to the working database

1 Launch AssetCenter.
2 Connect to your working database.

Creating an action to delete obsolete instances

1 Display the list of actions via the **Tools/Actions/Edit** menu.
2 Click **New**.
3 Populate the following fields:

Name	SQL name	Value
Name	Name	Delete finished instances
Context	ContextTable	Workflow schemes (amWfScheme)
Type	seActionType	Script
SQL name	SQLName	DeleteFinishedWfInstances
Script of the action	Script	See (*) below.

(*) Script of the action:

```
Const NumberOfInstanceToDelete = 50

Dim lRc As Long
Dim i As Long

i = 0

If [AutoCleaningDelay] >= 0 Then
  Dim hqWfInstance As Long
  hqWfInstance = AmQueryCreate()
  lRc = AmQueryExec(hqWfInstance, "SELECT lWfInstanceId FROM amWfInsta
nce WHERE lWfSchId = "& [lWfSchId] & " And seStatus = 1 AND ADDSECONDS
(dtCompleted, " & [AutoCleaningDelay] & ") < GetDate()" )

  Do While (lRc = 0 And i < NumberOfInstanceToDelete)
    Dim hrWfInstance As Long
```

```
    hrWfInstance = AmGetRecordHandle(hqWfInstance)
    lRc = AmDeleteRecord(hrWfInstance)
    lRc = AmReleaseHandle(hrWfInstance)
    lRc = AmQueryNext(hqWfInstance)
    i = i + 1
  Loop
End If
```

4 Click **Create**.

Creating a workflow scheme to automate the execution of the action

1 Display the list of workflow schemes via the **Tools/Workflow/Workflow schemes** menu.
2 Click **New**.
3 Populate the following fields:

Name	SQL name	Value
Name	Name	Delete finished workflows
Reference	Ref	ADM_CLEAN_WF_INSTC
Context of start object	StartContextTable	Workflow schemes (amWfScheme)
Execution group	GroupName	Specify any name in order to automate the execution of the workflow scheme in AssetCenter Server (**ADMIN**, for example).
One single active workflow instance for an object	bUniqueActive	Select this option.
Do not save instances in the database	bTransient	Do not select this option

4 Click **Create**.
5 Select the **Activities** tab.
6 Right-click on the **Start** activity and select the **Add an event** menu.
7 Populate the following fields:

Label	SQL name	Value
Name	Name	Timer
Type	seMonitoringType	Periodical
Reinitialize workflow instance if there is already one in progress	Periodical	Do not select this option

Label	SQL name	Value
AQL condition	AQLCond	`AutoCleaningDelay >= 0`

8 Select and populate the **Parameters** tab according to your needs.

9 Click **Add**.

10 Right-click and select the **Add an activity** menu.

11 Populate the following fields:

Label	SQL name	Value
Name	Name	Clean W/F instances
Type	seType	Automatic action
Log task	bLogWorkItem	Select this option.
Context	ContextTable	Workflow schemes (amWfScheme)
Input condition	seInCond	OR

12 Click **Add**.

13 Select the **Parameters** tab.

14 Populate the following fields:

Label	SQL name	Value
Execute actions immediately	bExecImmediately	Select this option.
Actions	Actions	Delete finished workflows

15 Click **Close**.

16 Double-click the activity **Clean W/F instances**.

17 Right-click on the **Executed** event and select the **Detail of event** menu.

18 Populate the following fields:

Label	SQL name	Value
Processing	seProcessingMode	Log event and process immediately

19 Click **Close**.

20 Using the mouse, create a link between the **Start** and **Clean W/F instances** activities.

21 Right-click the **Executed** event and select the **Terminal event** menu.

22 Click **Modify**.

Configuring AssetCenter Server to automate the execution of the workflow scheme

1 Launch AssetCenter Server.

2 Configure the module that will trigger the execution of the **Delete finished workflows** workflow via the **Tools/Configure modules** menu.

It is the module called **Execute workflow rules for execution group 'X'**, where **X** is the value of the **Execution group** field (GroupName) defined at the level of the workflow scheme.

3 Leave AssetCenter Server active if you want the workflow to execute automatically.

9 Exporting data and creating SQL views

CHAPTER

This chapter explains how to export AssetCenter data and manage SQL views of the database.

Definitions

Export scripts

The export scripts enable you to export data or (re)create/delete SQL views via AssetCenter Export or **amexp.exe**. You can save the export scripts that you define in order to reuse them.

An export script runs either:

- in "Export Mode" for exporting data.
- or in "Views Mode" to (re)create or delete SQL views from the database.
- Actions to perform, for creating/deleting SQL views.

An export script runs either:

- In "Export data" mode for exporting data.

- Or in "Create/Drop SQL views" mode to (re)create or delete SQL views from the database.

Export queries

You define export queries using AssetCenter Export.

An export query is defined by:

- A name.
- An eventual export file (when using the "Export data" mode).
- A comment (which is not exported).
- A starting table.
- A list of columns to be extracted (fields, links, features and calculated fields from the starting table) and associated sort criteria.
- A filter containing the WHERE clause and defining the extraction conditions.
- A filter containing the HAVING clause and defining the extraction conditions.
- The text of the query (corresponding to the **Filter (WHERE clause)** and "HAVING Clause") tabs.
- A preview tab.

Exporting data from the AssetCenter database

You can export data from the AssetCenter database to text files:

- Using an export script.
- Via the **Export the list** shortcut menu item. This menu item is displayed when at least one list or tab list is displayed. It allows you to export the active list.

Exporting data using an export script

1. Start AssetCenter Export using the **Start** menu or from the AssetCenter program group.
2. Define an export script whose mode is set to "Export data" mode:
   1. In the **Queries** tab, write the queries defining the data to export.

2 In the **Formatting** tab, specify the format for the text files to which the data will be exported.

3 Use the **File/ Save script** or **File/ Save script as** menu item to save the script.

3 Execute the export script:

- Either directly in AssetCenter Export using the <Execute script>
- Or by launching **amexpl.exe** in DOS.

Note:

To preserve the coherency of the access restrictions that you defined in AssetCenter, you can only launch AssetCenter Export or **amexpl.exe** as an administrator ("Admin" or user with administrative rights).

Exporting data using the Export the list shortcut menu item

The **Export the list** shortcut menu item is accessible to all AssetCenter users; it allows users to export the data they are authorized to view.

1 Display the list you want to export (main list or tab list). If several lists are displayed on the screen, make sure you are in the list to export.

2 Select the **Export the list** contextual menu item.

3 Populate the window that appears, then click the **Export** button.

Note:

For further information on the **File/ Export** menu item, refer to the **User interface** guide, chapter "First steps with AssetCenter", section "Record lists", sub-section "Exporting a list".

Managing SQL views in the AssetCenter database

AssetCenter Export allows you to create, recreate or delete SQL views in the AssetCenter database. External tools can then use these views instead of text files.

Note:

Warning: The SQL views that the export scripts allow you to create/modify/delete are different from views in the AssetCenter sense of the term. A SQL view is equivalent to the SQL "CREATE VIEW" statement.

To create, recreate or delete SQL views in the AssetCenter database:

1 Start AssetCenter Export.
2 Define an export script whose mode is set to "Create/Drop SQL views":
   1 In the **Queries** tab, write the queries defining the data to extract.
   2 In the **Views** tab, specify the actions to perform: create, modify or delete views, directly execute the resulting SQL script, or save to a file.
   3 Save the export script.
3 Execute the export script.
   - Either directly in AssetCenter Export.
   - Or by launching **amexpl.exe**.

Recommendations

We advise against using the "Id" fields of tables as reconciliation keys if you want to reimport data you have exported. In effect, the corresponding identification numbers are not fixed and can be subject to modification. Use keys whose values are "changeless" such as the AssetTag of assets.

Defining an export script

To export data or generate SQL views for your database, you must define the export scripts and export queries they contain. To do this, use AssetCenter Export.

An export script runs either:

- in "Export Mode" for exporting data.
- or in "Views Mode" to (re)create or delete SQL views from the database.

Actions to perform, for creating/deleting SQL views.

An export script runs either:

- In "Export data" mode for exporting data.
- Or in "Create/Drop SQL views" mode to (re)create or delete SQL views from the database.

This section explains how to create export scripts:

- Methodology
- Defining export queries
- Output format of an export script
- Actions concerning SQL views

Methodology

To create or modify an export script:

1. Start AssetCenter Export.
2. Open the appropriate database. Warning: You can only connect using the "Admin" login or a login with administration rights.
3. Create a new script using the **File/ New script** menu item, or open a script to be modified using the **File/ Open script** menu item.
4. At the top of the AssetCenter Export screen, define if you want to export data (Export data mode) or manage SQL views (Views mode) of the database.
5. Write the queries of the export script in the **Queries** tab.
6. If you export data, specify the output format for the exported data in the **Formatting** tab.
7. If you want to manage SQL views, specify what you want to do in the **Views** tab.
8. Save the script using the **File/ Save script** menu item or **File/ Save script as**.

Defining export queries

You can define the queries of an export script in the **Queries** tab of AssetCenter Export.

- Click the **New** button to add an export query.
- Click the **Delete** button to delete a selected export query.

Create a query in an export script

1 Click **New** in the **Queries** tab.

2 Define the query name. This name is used in the execution log in the **Messages** tab of the export script detail.

3 You can enter comments (they are not exported).

4 Define the data to be extracted in the **Query** field.

5 If you want to export data rather than create/modify/delete views, then in the **File** field specify the path and name of the output text file to which the exported data selected by the query will be written. Thus an export script containing several export queries will generate several text files.

Note:

The **File** field is not displayed if you selected the **Create/Drop SQL views** option.

Data to be extracted

To indicate the data to be extracted, populate the **Query** field in the detail of the export script query. The query applies to a table in the AssetCenter database.

You can enter the query directly, or click the button to access a window that will help you define the query:

Columns to be exported and sort order

In the **Columns and sort** tab, you define the list of fields, links, features and calculated fields to be exported, as well as the associated sort criteria.

Select one-by-one all the fields, links, features and calculated fields used for the export from the list on the left, and click the arrow to insert them in the list on the right.

For each line in the list on the right:

- Select the **Visibility** check box in order to export the column. If the **Visibility** box is not checked, the column is not exported (it may be used, however, to sort exported data, etc.).
- Select the **Group by** check box to group data by the field corresponding to the column. This is equivalent to adding the "GROUP BY <Field name>" clause to your SQL query.

For example:

```
SELECT Brand, Count(lModelId) FROM amModel GROUP BY Brand ORDER BY Bra
nd
```

Warning:

If you check the **Group by** box, the "GROUP BY" is appended to the query, but in order for the query to be valid, you must also add the appropriate aggregate functions in the SELECT clause.

Define the sort order for the exported data:

1 In order to define a sort by index, select an index in the **Sort by index** field.

2 Otherwise, check the appropriate **Sort** boxes in the desired sort order.

Note:

Except when using SQL Anywhere, you can check the **Force indexes** option in order to force the use of indexes referenced in the query. For further information, please refer to the manual entitled "Reference Guide: Administration and Advanced Use", chapter "Writing queries in AQL", section "Sorts and indexes".

If you check the **Unique records only** option, lines that are absolutely identical are only exported once. This is equivalent to adding the "DISTINCT" clause to the SQL query.

Example with no check in the **Unique records only** box:

```
SELECT Brand FROM amModel
```

Example with a check in the **Unique records only** box:

```
SELECT DISTINCT Brand FROM amModel
```

Filters

You can define two types of filters for selecting the data to be extracted:

- An AQL query using the WHERE clause in the **Filter (WHERE clause)** tab.
- An AQL query using the HAVING clause in the **HAVING Clause** tab.

Displaying the query

The AQL query that you define in the **Columns and sort**, **Filter (WHERE clause)** and **HAVING Clause** tabs is displayed in the **Queries** tab.

Previewing the query results

You can test the query and view it in SQL language syntax from the **Preview** tab.

Simply click  to preview the query results as a list of records. Note that AssetCenter displays the number of records matching your query at the bottom right of the window.

Output format of an export script

If you select the **Export data** option, you can define the format for the output text files in the **Formatting** tab. This format is applied to all export queries.

Note:

The **Formatting** tab is not displayed if you choose to drop (delete)/ create/ recreate views.

Column title

Select a value if you want the first line of the export file to include:

- The alias of the columns specified in the export query.
- The "SQL name" of fields or links corresponding to columns.
- The "Description" of fields or links corresponding to columns.

Column separator

This separator is inserted between the data in each column.

Text identifier

The identifier surrounds text strings. If you use the ' character, any ' characters you export will be output as ''. And vice-versa for the " character.

Character set

This option allows you to choose either the ANSI, OEM(DOS), UFT-8, UNICODE or Latin 1. character set.

Decimal separator

This character is used to separate the decimal part of exported numbers.

Date separator

This character is inserted between the day, the month and the year of exported dates.

Date format

The date format defines the order in which days (DD), months (MM) and years (YY) are exported.

Year format

Defines whether years will be exported with 2 or 4 digits.

Time separator

This character is inserted between the hours, minutes and seconds.

Display seconds

Indicate if you want seconds to appear in exported times.

Actions concerning SQL views

If you want to delete or (re)create SQL views corresponding to export queries, you can use the **Views** tab to define the actions to be performed.

Note:

The **Views** tab is not displayed if you choose the **Export data** option.

Select one of the actions to perform in the "Actions" frame:

- Create or recreate views.
- Delete views.

In the "SQL view-manipulation script" section, specify how you want the query to be processed (**Queries** tab, **Actions** frame):

- To create or recreate SQL views directly when executing the export script, select the **Execute SQL directly** option.
- To generate a SQL view script to create a view ("CREATE VIEW" statement) or drop a view ("DROP VIEW" statement), select the **Save the SQL code to a file** option, then:
  1 Click the button to indicate the name and path of the file.
  2 Specify a SQL statement separator: ";" (for Oracle) or "GO" (for all other DBMSs).

Executing an export script

You can use export scripts to export data or manage SQL views.

This section describes two methods for executing an export script:

- Executing an export script from AssetCenter Export.
- Executing an export script from DOS.

Executing an export script from AssetCenter Export

To execute an export script from AssetCenter Export:

1 Start AssetCenter Export.
2 Define your export script and save it.
3 Then execute the script:
  - Either by using the **Actions/ Execute the script** menu item.
  - Or by pressing F8.
  - Or by clicking .

Information about the progress of the export process is displayed in the **Messages** tab.

If the export process terminated successfully, the last message displayed is: "Script successfully executed ". If an error occurs, the following message is displayed: "An error occurred while executing script".

An icon is displayed before all messages:

General information.

Error.

Export was successful.

⚠ Warning.

Executing an export script from DOS

How it works

In order to execute the DOS software "online", you must first create an export script using AssetCenter Export.

You can execute, either manually or automatically (with a batch file, for example), an export command using the **amexpl.exe** program in the **bin** sub-folder of the AssetCenter installation folder.

Syntax

```
amexpl32 [-verbose] [-?|h|H] -script:<script>
-cnx:<cnx> [-login:<login>]
[-password:<password>]
```

-verbose: displays messages during the export process.

-?, -h or -H: displays help messages.

-script: path and name of the export file to execute.

-cnx: name of the connection to the AssetCenter database (as it appears in the **File/ Manage connections** menu item).

-login: login name of the database administrator ("Admin" or a user with administrative rights).

-password: password for the login.

Strings between <> cannot include spaces.

For example:

```
amexpl32 -verbose -script:ibmassets.scx -cnx:GeneralDatabse -login:Gerald
 -password:PAssword
```

10 Scripts

CHAPTER

This chapter explains how to use scripts.

Definition of a script

Overview

The word "script" generally designates a program written in a high-level language. In AssetCenter, this notion includes three types of scripts:

- Procedural scripts, which include:
  - Calculation scripts written in Basic used to calculate the values of fields, determine the properties of objects in the AssetCenter database, etc.
  - Basic scripts executing tasks, particularly in actions.

Note:

These Basic programs can incorporate functions. This type of script is the subject of this chapter.

- Declarative scripts. These are import and export scripts that use their own scripting language, different from Basic. This type of script is documented in full detail in the manuals entitled "Reference Guide: Administration and Advanced Use", chapter "Importing data" and "Reference Guide: Administration and Advanced Use", "Exporting data and managing SQL views"
- "Mixed", both declarative and procedural. This type of script is used in the wizards in AssetCenter.

Information about this version of Basic

The version of Basic used in AssetCenter is a sub-set developed by Cypress compatible with "Visual Basic for ApplicationsTM". Please refer to the Basic documentation for additional information on this language, its structure and syntax.

Only certain Visual Basic for Applications functions are supported, e.g.:

- File access functions are not supported.
- There is limited support for date and time functions.

  This is particularly true in UNIX.
- The Visual Basic for Applications controls are not available.

Note:

To consult the Programmer's Reference of a Basic function or keyword, position the cursor on the word and press F1: Contextual help is displayed.

Data access notation

The Basic syntax used in AssetCenter is similar to standard syntax, except for data access functions from the current record; this uses the following format:

```
[Link.Link.Field]
```

Example from the Models table:

```
[Category.FullName]
```

Note:

You can use the following syntax to recover the ID number of a link:

```
[Link.Link]
```

When you want to refer to a link, you can use either the link's SQL name or the link's key name.

Example:

```
RetVal=[Contact.Location] or RetVal=[Contact.lLocaId]
```

Both examples return the same result, the ID of the link.

Applications of scripts

AssetCenter allows you to associate the following properties with a Basic "Script":

- For configuring the default values of fields (**Configure object** command in the shortcut menu).
- For the default value of a feature associated with a table.
- In Basic type calculated fields.
- For configuring fields (**Configure object** command in the shortcut menu or AssetCenter Database Administrator):
  - **Default value**.
  - **Mandatory nature**.
  - **History**.
  - **Read-only**.
- For the parameters of a feature associated with a table:
  - **Default value** (SQL name: DefValScript).
  - **Available** (SQL name: seAvailable).
  - **Force display** (SQL name: seForceDisplay).
  - **Mandatory** (SQL name: seMandatory).
  - **Keep history** (SQL name: seKeepHistory).
- For **Script** actions:
  - **Action script** (SQL name: Script) for a **Script** action.
- In the wizards:

    - Start and end of wizard scripts.
    - Scripts for defining the node properties.
- In "Basic" calculated fields.
- In the workflow:
    - For **Test / script** workflow activities.
    - For **Base** workflow events.
    - For **Calculated individual** workflow assignees in the amWfOrghole table.

Introduction to functions

This section includes information on the following:
- Definition of a function
- Built-in functions and programmable functions
- Function and parameter types

Definition of a function

A function is a program that performs operations and returns a value to the user. This value is called the "return value" or "return code".

Functions have the following structure:

```
Function <Function name> (<Parameter> As <Parameter type>[, ..., <Paramet
er> As <Parameter type>]) As <Function type>

<Program (script) executed by the function. This program must define the
return value.>

End Function

End Function
```

This structure applies to both built-in functions and programmable functions.

Built-in functions and programmable functions

Built-in functions and programmable functions are the two major function categories available under AssetCenter.

Built-in functions

Built-in functions are similar to software items that have already been written for the user. These software items perform all types of tasks (calculations, conversions of data provided by the user) and return a result. The user simply calls the function by its name and provides any information required to return a result. The items of information provided by the user are called the "parameters".

For example, the **AmConvertCurrency()** function converts an amount in currency A to an amount in currency B, using an exchange rate defined at a given date. In this example:

- The function name is **AmConvertCurrency**
- The parameters the user must provide are:
  - Currency A
  - Currency B
  - The amount to convert
  - The date when the conversion will take place (used to identify the conversion rate to use).

This function performs the conversion, and provides a return value corresponding to the result of the conversion.

Programmable functions

Programmable functions are software items users can write themselves. The user is responsible for explicitly defining the value to be returned in the **RetVal** variable (also called the "return value") by the programmable function, in the following format:

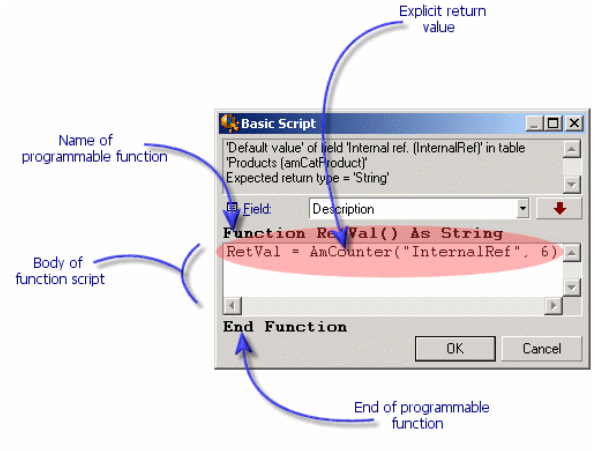
```
RetVal=<Expression>
```

Note:

AssetCenter refuses to compile the script of a function for which the return value is not defined.

Programmable functions are accessible through a script builder (by clicking the button in a scriptable field). The script builder is designed to help users create the software item corresponding to a function. The script builder includes a template for writing programmable functions:

Figure 10.1. Script - builder



A description the programmable function is available at the top of the script builder window. It identifies the object concerned by the function (for example, the default value of the **Bar code** field (SQL name: BarCode) in the table of assets) as well as the expected return type code (using the previous example: "String").

Function and parameter types

Function types

The type of a built-in function is the type of the value returned by the function. We suggest you pay special attention to this point, because it can cause compilation and execution errors in Basic scripts.

For example, you cannot use a function that returns a value of one type when defining the default value of a field of a different type. For example, try to assign this default script to any "Date" or "Date and time" type field:

```
RetVal=AmLoginName()
```

The **AmLoginName()** function returns the name of the connected user, in the form of a character string (String type). This return value is therefore in a format incompatible with the format of a "Date" field, and AssetCenter will

display an error message the next time that you create a record in the same table.

Parameter types

The parameters used in built-in functions also have a type; you must respect that type for the function to execute correctly. If an error occurs in the type of a parameter, AssetCenter displays an error message when executing the function.

List of types

The following table summarizes the various types available for a function or a parameter:

Table 10.1. Functions/parameters - types

Type	Description
Integer	Integer from -32,768 to +32,767.
Long	Integer from -2,147,483,647 to +2,147,483,646.
Double	8 byte floating-point number.
String	Text in which all characters are allowed.
Date	Date or Date+Time.
Variant	Generic type that can represent any type.

Determining the return type of a programmable function

Before editing a script, you should determine the function involved and its type. This information is displayed in bold in all "Basic script" windows in the following format:

```
Function <Function name>() As <Function type>
```

The three most common function types are "Boolean", "Integer" and "String":

- "Boolean" functions return either "TRUE" or "FALSE"; any other value causes an error when compiling the Basic script.
- "Integer" functions return only integer values (for example. 0, 1, 8, 12).
- "String" functions return only character strings (for example: "Building21") between quotes.

Note:

If you do not respect the type of a function, you may obtain errors when compiling the Basic program. Always note the type of the function you are using.

The function name and type allow you to determine the return code you must use in the script, in the following format:

```
RetVal=<Expression respecting the function type>
```

Classifying Basic functions

The Basic used in scripts uses various classes of functions:

- Traditional Basic functions following the "Visual Basic for Applications TM" standard
- Generic functions specific to AssetCenter, which may be used in all places where scripts are used.
- Specific functions, which may be used in certain parts of AssetCenter.

First steps in writing scripts

This section deals with the functioning of scripts and includes an example scenario:

- Example scenario
- Step 1- Create the Tutorial feature
- Step 2- Open the edit window
- Step 3- Analyze and define the algorithm
- Step 4- Draw up the Basic script
- Step 5- Test the Basic script

Example scenario

Objective

To make sure that the "Tutorial" feature is only available for the "Computer/ Motherboard/" model and its descendants.

Method

Attaching a Basic script to the **Available** parameter (SQL name: seAvailable) of the "Tutorial" feature.

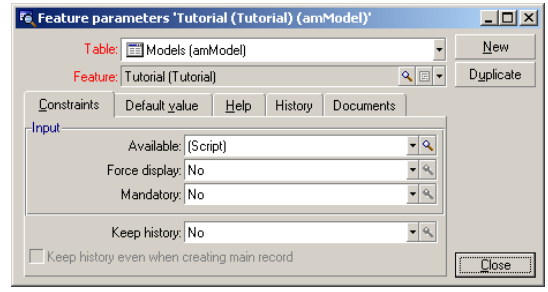
Step 1: Create the feature "Tutorial"

Select the **Administration/ Features** menu item. Click **New** to create a new feature. Populate this feature as shown below:

Field name	Value
Title (SQL name: TextLabel)	"Tutorial"
SQL name (SQL name: SQLName)	"Tutorial"
Input type (SQL name: seDataType)	**Numerical**

Click **Create** to create the feature.

Go to the **Parameters** tab and click ⊞ to edit the parameters of the "Tutorial" feature. Populate the **Constraints** tab as shown below.



Step 2: Open the edit window

In the **Constraints** tab, set the **Available** parameter (SQL name: seAvailable) to "(Script)". Click the magnifier button. AssetCenter opens the script edit window:

Step 3: Analyze and define the algorithm

The algorithm must fulfill the following tasks:

- Set the **Available** field (SQL name: seAvailable) to **Yes** if the model is "/Computer/ Motherboard/" or one of its descendants.
- Set the **Available** field to **No** in all other cases.

Our algorithm is therefore as follows

```
If the model's full name starts with "/Computer/Motherboard" Then
The feature is available
Otherwise
The feature is not available
```

It is therefore the value of the **Full name** field (Nom SQL : FullName) in the Models table that determines the value of the **Available** field of the feature. Only this field appears in our algorithm.

Click the magnifier button next to the **Available** field to start editing the Basic script. The drop-down list in the edit window lets you select the **Full name** field (SQL name: FullName) from the Models table.

Once you have selected the field, transfer it to the edit window by clicking the button.

Step 4: Draw up the Basic script

You now need to write the algorithm from step 3 in Basic in the edit window established for this task.

```
If Left([FullName], Len("/Computer/Motherboard/"))="/Computer/Motherboard
/" Then
   RetVal=1
Else
   RetVal=0
End If
```

Note:

Scripts are not case-sensitive.

Click **OK** to confirm your script.

Step 5: Test the Basic script

This step allows you to make sure that the script functions correctly.

1. Open the Models table by selecting the **Portfolio/ Models** menu item. Click **New** to create a new model.
2. Populate the mandatory fields only.
   1. **Name**
   2. **Sub-model of** (SQL name: Parent) with "Computer/Motherboard".
   3. **Nature** (SQL name: Nature)
   4. **Bar code** (SQL name: BarCode).
3. Click **Create** to create your new model.
4. Select the **Features** tab and click to add a feature. The selection screen shows the name of the feature for which you have just edited the script.
5. Change the value of the **Sub-model of** field to "/Computer/" and validate this change by clicking **Modify**.
6. Select the **Features** tab in the model detail and click the button to add a feature. The selection screen no longer shows the name of the feature for which you have just edited the script.

The script fulfills its function correctly.

Script library

AssetCenter enables you to save script libraries in order to centralize the access to these scripts.

You can open the script library via the **Administration/ Script library**.

The recorded script libraries are called up by the **amEvalScript** API command.

For more information about the **amEvalScript** API, refer to the "Programmer's Reference" guide, section "Alphabetic reference".

Concepts

In AssetCenter, a script defines a function.

Creating a script library thus implies that you will define a set of functions.

Creating a script library

To create a script library:

1 Open the list of script libraries.

2 Populate the **Name** field with the name of your library.

3 Enter your script in the **Script** field.

4 Validate your script by clicking **Create**.

For example, create the "lib" library by entering the following script:

```
function FullName(strName As String, strFirstName As String) As String
  FullName = strFirstName & ", " & strName
end function
```

This function returns a string composed of an employee's first and last name.

Warning:

Each created function must have a different name for the set of script libraries created.

Calling up a script in a script library

To call up a script from a library, you must define several parameters: the name of the library, the function defined in the script, and the parameters associated with this function.

For example, create a script-type action, "callEvalScript", that will use the previously created library.

1 Populate the **Context** field (SQL name: ContextTable) with the Departments and employees table (SQL name: amEmplDept).

2 Enter the following script in the **Script** tab:

```
Dim strFullName As String
strFullName = amEvalScript("lib", "FullName", "", [Name], [FirstName])
amMsgBox (strFullName)
```

This script calls up the "FullName function from the "lib" library and displays the first and last name of the employee in the dialog box.

3 Validate this by clicking **Create**.

Note:

The context parameter, normally used with the **amEvalSript** API is not used in the case where you call on a script library.

Tips and warnings

This section includes several tips that may help you when writing scripts.

Precautions when using programmable functions

Here a few precautionary measures you should remember when writing your scripts:

- The purpose of programmable functions, such as that which defines the default value of a field or a link, is to set the return value of the function. Therefore you are strongly advised against performing other operations within a programmable function. In the best case scenario, you may note a general performance hit, and in the worst case, you can damage your database.
- Programmable functions are widely used in AssetCenter. When possible, try to optimize your scripts to maintain overall AssetCenter performance.

Format for "Date+Time" constants in scripts

Dates referenced in scripts are expressed in international format, regardless of the display options defined by the user:

yyyy/mm/dd hh:mm:ss

Example:

```
RetVal="2001/07/12 13:05:00"
```

Note:

You can also use a hyphen ("-") as a date separator.

"Basic" date

In Basic, a date can be expressed in international format, or as a "Double-precision number". In this case, the integer part of the number represents the number of days elapsed since December 30, 1899 at midnight, the decimal part represents the fraction of the current date (The number of seconds elapsed since the start of the day divided by 86400).

"Unix" date

Dates are expressed differently in Basic and under Unix:

In Unix, dates are expressed as an "Integer (32 bit)" that represents the number of seconds elapsed since January 1, 1870 at midnight, independent of time zones (UTC time).

Format for "Duration" constants in scripts

In scripts, durations are stored and expressed in seconds. For example, to set the default value for a "Duration" type field to 3 days, use the following script:

```
RetVal=259200
```

Likewise, functions that calculate durations, such as the **AmWorkTimeSpanBetween()** function, return a number of seconds.

Note:

For conversions, AssetCenter considers a year as being 12 months and a month being 30 days (thus 1 year = 360 days).

Read and write access to a system itemized list

AssetCenter manages system itemized lists by assigning an integer to each possible value in the itemized list.

Let's take the example of the itemized list used to populate the **Assignment** field (SQL name: seAssignment) in the **Standard assignment** zone of the **General** tab of an asset detail.

The following table summarizes the values used in this itemized list:

Value in itemized list	Associated integer
In use	0
In stock	1
Retired asset	2
Awaiting delivery	3

Thus, in order to define the default value of an itemized list, you need to:

1 Identify the integer corresponding to the appropriate value

2 Edit the following string:

```
RetVal=<Integer associated with the appropriate value>
```

Using this example, if you want to set the default value of the system itemized list used in the **Affectation** field to **Awaiting delivery**, you need to edit the string as follows:

```
RetVal=3
```

Note:

Do not confuse a system itemized list with a user-defined closed itemized list.

Note:

The full list of system itemized list values is included in **database.txt**, which can be found in the **doc\infos**" sub-folder of the AssetCenter installation folder. The two columns, "Data display and entry type" and "Additional information on data display and entry type", describe the itemized list type and the values taken by an itemized list respectively.

The "CurrentUser" virtual link

Definition

"CurrentUser" may be considered as a link starting in all tables and pointing to the record in the table of departments and employees corresponding to the current user.

- In the "CurrentUser" format, it points to the record corresponding to the current user, and returns the user's ID number.
- In the "CurrentUser.<SQL name of field>" format, it returns the value of the field for the current user.

Note:

This virtual link is not displayed in the list of fields and links; therefore it is not directly accessible in the script builder. You must enter this expression manually.

Equivalencies

The **AmLoginName()** and **AmLoginId()** functions, which return the current user's name and ID, respectively, may be considered as functions derived from "CurrentUser". In effect, the following are equivalent:

- AmLoginName()=[CurrentUser.Name]
- AmLoginId()=[CurrentUser.lPersId]

Commenting a Basic script

It is often useful to comment a Basic script to specify, in clear terms, what it performs or to allow a user to understand and to be able to modify the script. AssetCenter provides you with the ability to comment the body of a script using the apostrophe (') character. All the characters following a single straight quote mark on the same line are ignored by the compiler, which interprets them as comments. There are two possible situations:

- Either the comment has its own line in the Basic script, as shown below; No other precautions are necessary.

```
' Here we test the value of the Bar code field in the table of assets
' If this value is PC1, the return code is set to TRUE
If [BarCode]="PC1" Then
```

```
RetVal=True
End If
```

- Or the comment is added to the end of a line that must be interpreted by the Basic compiler. In this case, you must use the ":" character to separate the script from the comment. The comment remains prefixed by a single straight quote mark.

```
If [BarCode]="PC1" Then : ' Then BarCode is PC1
RetVal=TRUE : ' The return value is set to TRUE
End If : ' End of test
```

Triggering an error message

You can trigger an error message on purpose using the Err.Raise function. Its syntax is as follows:

```
Err.Raise (<Error number>, <Error message>)
```

Note:

When the creation or modification of a record is invalidated because of the value of the "Validity" field of the table concerned, it is good practice to trigger an error message with the Err.Raise function, in order to warn the user. Without this error message, the user will not necessarily understand why the record cannot be created or modified.

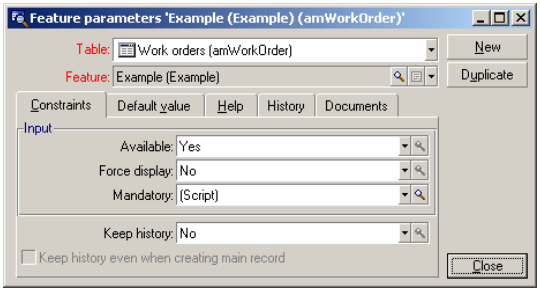
First example

This section deals with a hypothetical problem that can be solved using a Basic script. We recommend trying this problem yourself before consulting the described solution.

Description of the problem

A feature called "Example1", associated with the table of work orders, must be populated when work orders are closed. This feature is optional for work orders that are not closed. The rest of this example supposes that the feature has already been created, has an arbitrary input type, is associated with the

table of work orders, is available and is displayed by default (force display) as shown in the screen capture below:



Step 1: Analyze and define the algorithm

The algorithm must fulfill the following tasks:

- Set the **Mandatory** field (SQL name: seMandatory) to **Yes** if the ticket is closed.
- Set the **Mandatory** field (SQL name: seMandatory) to **No** in all other cases.

Our algorithm is therefore as follows

```
If the work order is closed Then
Populating the feature is mandatory
Else
Populating the feature is not mandatory
```

A work order is closed if its **Status** field (SQL name: seStatus) is either **Closed**.

It is therefore the value of the **Status** field (SQL name: seStatus) in the table of work orders that determines the value of the **Mandatory** field (SQL name: seMandatory) of the feature. Only this field appears in our algorithm.

The drop-down list in the edit window allows you to find the **Status** field in the table of work orders.

Once you have selected the field, transfer it to the edit window by clicking the  button.

This field is populated using a system itemized list. We therefore have:

Value in itemized list	Associated integer
Notified	0
Scheduled	1
In progress	2

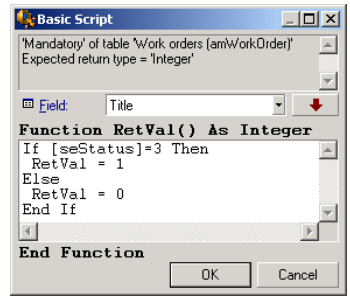
Value in itemized list	Associated integer
Closed	3

The value in the itemized list that interests us is therefore:

- **Closed** corresponding to the value "3"

Step 2: Draw up the Basic script

You now need to write the algorithm from step 1 in Basic.



Click **OK** to confirm your script.

Step 3: Test the Basic script

This step allows you to make sure that the script functions correctly.

1. Open the table of work orders by selecting the **Portfolio/ Work orders** menu item. Select a ticket that is **Closed** (or create a work order with this status if there isn't one available).
2. Select the **Features** tab. AssetCenter has added the feature to the work order concerned and the feature is mandatory.
3. Now select a work order with a status other than **Closed**. Go to the **Features** tab of this work order. The feature "Example1" is displayed but it is an optional field.

The script fulfills its function correctly.

Second example

This section deals with a hypothetical problem that can be solved using a Basic script. We recommend trying this problem yourself before consulting the described solution.

Description of the problem

By default, we want the **Field1** field (SQL name: Field1) in the detail of an employee show the name and first name of the employee if both of these exist, or just the name if the first name is missing.

Step 1: Analyze and define the algorithm

The algorithm must fulfill the following tasks:

- Display by default the name and the first name of the employee in **Field1** (SQL name: Field1) in the employee detail if both the name and first name exist.
- Display by default the name alone of the employee in **Field1** (SQL name: Field1) in the employee detail if the first name does not exist.

Our algorithm is therefore as follows:

```
If the first name of the employee does not exist Then
The default value of "Field1" is the name of the employee
Else
The default value of "Field1" is "Last name," "First name"
```

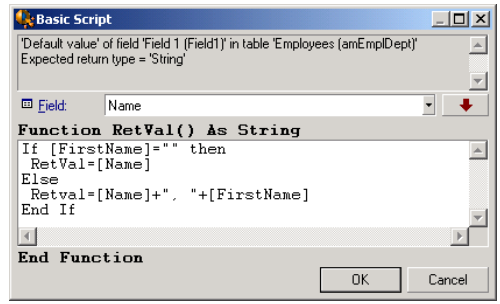
It is therefore the values of the **Name** (SQL name: Name) and **First** (SQL name: FirstName) fields in the table of employees which determine the default value of **Field1** (SQL name: Field1) the employee detail. Only these two fields appear in our algorithm.

Right-click **Field1** (SQL name: Field1) in the employee detail and select **Configure object**.

Click the magnifier button  next to the **Default** field to edit the Basic script.

Step 2: Draw up the Basic script

You now need to write the algorithm from step 1 in Basic.



Click **OK** to confirm your script.

Step 3: Test the Basic script

This step allows you to make sure that the script functions correctly.

1. Open the table of employees by selecting the **Portfolio/ Departments and employees** menu item and create a new employee.
2. Populate the **Name** (SQL name: Name) and **First** (SQL name: FirstName) fields, then click **Create** to confirm. AssetCenter displays the name and first name of the employee in **Field1** (SQL name: Field1).

The script fulfills its function correctly.

11 Calendars

CHAPTER

Use the Tools/ Calendars menu item to access the list of calendars.

Overview of calendars

The calendar detail

The detail of a calendar shows:

- General information that enables you to uniquely identify the calendar:
  - The **Name** (SQL name: Name) of the calendar.
  - The **Time zone** (SQL name: TimeZone) to which the calendar is attached.
- The description of usual business hours in the **Timetables** tab.
- The list of exceptions to this timetable in the **Exceptions** tab.
- A preview of business hours for a given period that takes into account the rules defined in the **Timetable** and **Exceptions** tabs.

Using calendars

Calendars are associated with:

- Workflow activities.

They allow you to set the time when alarms defined in escalation schemes or in workflow activities are to be triggered. The **Time limit** defined in these alarms is, in effect, specified in business hours.

Warning:

If you modify a calendar in the database, modifications are only taken into account at the level of those fields linked to the calendar when you exit and relaunch AssetCenter.

Impact of calendars on certain areas of functionality

Calendars have an impact on certain areas of functionality of AssetCenter. Modifying a calendar brings about changes both directly and indirectly to certain records in the database. Calendars are involved in:

- The execution times of workflow tasks.
- Alarms associated with workflow activities.

Methodology used to create a calendar

Use the following steps to create a calendar:

1. Start by identifying the calendar by giving it a **Name** (SQL name: Name).
2. If necessary, associate the calendar with a time zone by populating the **Time zone** field (SQL name: TimeZone).
3. Define the daily business hours in the **Timetables** tab of the calendar detail.
4. Next, define any exceptions to these business hours in the **Exceptions** tab of the calendar detail.
5. Finally, you can verify the functioning of the calendar via the **Preview** tab.

Description of how to create a calendar

A calendar is created step by step:

1 Entering general information
2 Populating the Timetable tab
3 Populating the Exceptions tab
4 Checking the calendar

Entering general information

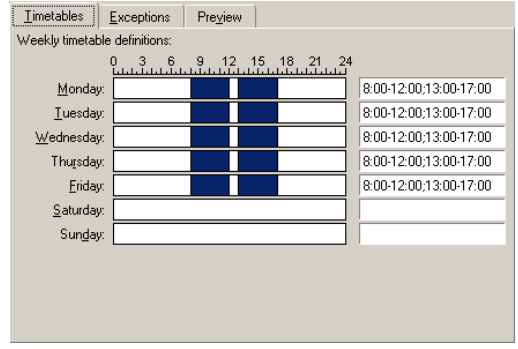
Before entering the actual business hours and exceptions, you must identify the calendar by populating the **Name** field (SQL name: Name) in the detail screen.

You can also associate the calendar with a time zone by populating the **Time zone** field (SQL name: TimeZone).

Populating the Timetable tab

The **Timetables** tab in a calendar detail defines the weekly timetables associated with this calendar. The periods of business hours defined in this tab define the weekly timetables associated with this calendar. These periods describe the general rule. Public holidays and the like constitute exceptions and are defined in the **Exceptions** tab.

Figure 11.1. Calendar - Timetables tab



For each day of the week, you can define one or more timetable periods representing business hours. You can define these using two methods:

- Graphically by using the graduated slider controls representing each day of the week.
  1. Click the control at the start of the timetable period.
  2. Extend the selection by dragging the mouse to the end of the timetable period. AssetCenter automatically populates the text field to the right of the graduated control.
  3. Repeat as necessary.
- "Manually", using the text field. Use the following syntax for this field:

```
<First hour of the business period>-<Last hour of the business period>;<
First hour of the business period >-< Last hour of the business period >;
...
```

The following format is used for time values:

```
<hh:mm[{AM|PM}]>
```

If the optional [AM|PM] parameter is not defined, AssetCenter considers that the 24 hour format is used.

AssetCenter automatically populates the graduated slider control situated to the left of the text field.

Note:

Using the graphical control only gives you a precision to the nearest half-hour. Using the text control is accurate to the nearest minute.

Populating the Exceptions tab

The **Exceptions** tab in the calendar detail defines exceptions to the weekly business periods defined in the **Timetables** tab.

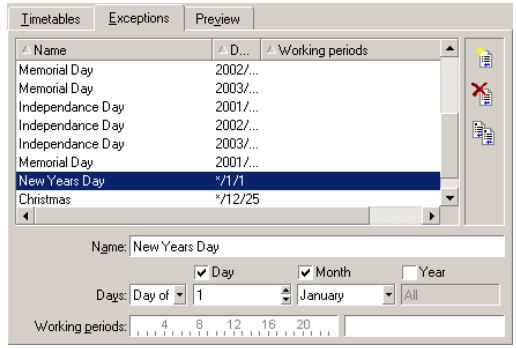
Methodology

To create an exception:

1. Give a "Name" to the exception.
2. Define the field of application of the exception by populating the **Days** field. Exceptions can be defined according to a given day, month or year.

3 It is also possible to define a business period within an exception using the **Working periods** field. This field allows you define more precise exceptions such as: "On the last Friday of each month, the team works from 8:30 A.M. to 10:30 A.M. and from 5:30 P.M. to 6:30 P.M".

The Exceptions tab



This tab is divided up into two parts.

- The first part gives you a list of exceptions and allows you to create, duplicate, destroy, modify, and cancel modifications using the buttons on the toolbar:
  - : Click this button to create a new exception.
  - : Click this button to delete an exception.
  - : Click this button to duplicate an exception.

Note:

The **Ranking** column allows you to sort exceptions by priority: It determines which exception takes priority in case of conflict. AssetCenter automatically assigns a ranking (from "P00" to "P15") to an exception. The smaller the number, the higher the priority of the exception. Thus an exception ranked "P06" takes priority over an exception ranked "P10".

- The second part gives you the details of the exception.

  The values taken by the **Days** field define the context of application of the exception:

Value of the Days field	Context of application of the exception
"Daily"	The exception applies to all days of the year without exception.
"Day of the year:"	The exception applies to given day or selection of days, defined using the **Day**, **Month** and **Year** check boxes.
"The first"	The rule applies to the weekday defined via the **Day** check box, and for the month(s) and year(s) defined via the **Month** and **Year** check boxes. **Example** "The first" Friday of each month.
"The second"	The rule applies to the weekday defined via the **Day** check box, and for the month(s) and year(s) defined via the **Month** and **year** check boxes. **Example** "The second" Monday of the month of September.
"The next to last"	The rule applies to the weekday defined via the **Day** check box, and for the month(s) and year(s) defined via the **Month** and **year** check boxes. **Example** "The next to last" Wednesday of the month of November.
"The last"	The rule applies to the weekday defined via the **Day** check box, and for the month(s) and year(s) defined via the **Month** and **Year** check boxes. **Example** "The last" Tuesday of each month throughout 2000.

Example

Employees at Taltek have the following days off:

- The first Friday of each month is a day off.
- During August, employees at Taltek only work the morning from 8:30 A.M. to 12:30 P.M.

Rule n°1: The first Friday of each month is a day off.

1. Click **New** to begin the creation of the exception.
2. The exception applies the first Friday of every month for all years. The **Month** and **Year** check boxes are therefore unchecked, meaning that the exception is independent of the month and the year. The **Day** box is checked, since the exception only applied to Fridays.

3 To end: Set the **Days** field to: "The first".

Rule n°2: During August, employees of Taltek only work the morning from 8:30 A.M. to 12:30 P.M.

1 Click **New**.

2 This exception is only dependent on the month (applies to the month of August only). The **Day** and **Year** boxes are therefore unchecked, the **Month** box is checked and the corresponding value set to "August".

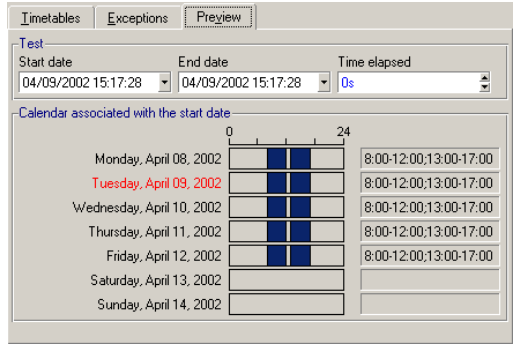
3 Since the exception only applies to a selection of days (all days throughout the month of August), the **Days** field just needs to be set to "Day of the year:".

4 During this period, employees work from 8:30 A.M. to 12:30 P.M. To finish entering the exception you just need to select the period from 8:30 A.M. to 12:30 P.M. in the **Working periods** field.

Checking the calendar

The **Preview** tab allows you to apply the rules defined in the **Timetables** and **Exceptions** tabs to a period selected using the **Start date** and **End date** fields in the **Test** zone.

Figure 11.2. Calendar - Preview tab



- The **Calendar associated with the start date** frame by default gives you a preview of the business hours during the week containing the selected "Start date".

- The **Time elapsed** field gives the total number of business hours during the selected period.

Note:

You can enter a duration in the **Time elapsed** field to force the recalculation of the **End date** field according to the given **Start date**.

12 Time zones

CHAPTER

This chapter explains how to use time zones.

Note:

This area of functionality is only available for certain AssetCenter license agreements.

Why manage time zones?

Since client machines and the database server can be separated geographically, AssetCenter manages time zones and time differences in relation to Greenwich Mean Time (GMT). AssetCenter respects the following rules:

- All "Date and time" type fields are displayed on the client machine respecting the time zone of the client machine.
- All "Date and time" type fields are stored on the server in reference to a defined time zone.
- All calculations involving dates and times take time zone differences into account.

Example

Let's take the example of a server situated in New York that has data indexed according the Paris (France) time zone and two client machines situated in London and in Paris. Let's start by defining the time zone of each of these client machines according to Greenwich Mean Time:

- Time zone of server = GMT-5
- Time zone of Paris client = GMT+1
- Time zone of London client = GMT
- Time zone of data = GMT+1

All "Date and time" type values are thus stored on the server in GMT+1 format and are displayed on the Paris client as GMT+1 and on the London client as GMT. For example, when taking a work order on the London client machine, if the resolution deadline of the work order is set to May 15, 2000 at 17:30, this is shown as follows on the other machines:

- On the server: May 15, 2000 at 12:30 P.M.
- On the Paris client: May 15, 2000 at 6:30 P.M.
- On the London client; May 15, 2000 at 5:30 P.M.

Implementing time zones

In order for time zones to be handled correctly under AssetCenter you need to follow the following steps:

1. Define the time zones when creating the database under AssetCenter Database Administrator using the **Use time zones** option.
2. Create the time zones (by importing the information relative to time zones, for example).
3. Define the time zone of your machine using the **Tools/ Change the time zone** menu item.
4. Define the calendars in accordance with the time zones.

Creating time zones

The time-zone functionality in AssetCenter, unlike in Windows, handles modifications to daylight-saving rules over the years. This enables you to

display local times in the past with greater accuracy. AssetCenter's use of time zone information allows you to:

- Display local dates and times taking into account daylight saving changes.
- Put yourself in the place of another location.

In order to avoid having to define time zones manually, AssetCenter ships with a description file containing the principal time zones. This file can be imported using the following procedure:

1 Select the **File/ Import** menu item. AssetCenter opens the import selection window.

2 Select "Execute a script" by clicking  . AssetCenter opens the database update screen. Click  to select the script to execute, in this case **tz.scr** in the **datasys** sub-folder of the AssetCenter installation folder.

3 Click **Import**. AssetCenter performs the import according to the script.

Managing a time zone

In this section, we will look at the **Daylight time** field (SQL name: memDaylightInfo) in more detail.

Format of Daylight time field

The **Daylight time** field (SQL name: memDaylightInfo) is structured as follows (in one single line):

```
<Year>=<DaylightInfo>|<Year>=<DaylightInfo>|<Year>=<DaylightInfo>|...
```

In the rest of this section, the following conventions will be used:

- <Year>=<DaylightInfo> together is called the "parameter"
- <Year> and <DaylightInfo> separately are called "arguments"

The table below gives you an overview of daylight savings changes according to the values of the <Year> and <DaylightInfo> arguments.

	The <DaylightInfo> argument is empty	The <DaylightInfo> argument has a value
The <Year> argument is empty. ("<Year>=" does not appear)	There is no change for daylight savings for the entirety of this time zone.	Daylight savings information is valid for all years, excepting those defined by parameters having a <Year> argument.
The <Year> argument has a value	Not applicable	Daylight savings information for this time zone is valid for every year from the year specified by the <Year> argument up until the next <Year> argument.

Values of the <Year> argument

The <Year> argument that specifies the year from which the daylight saving information defined in the <DaylightInfo> is applicable, can take any four figure year value (e.g. 1990, 1997, 1998, 2012).

Values of the <DaylightInfo> argument

The full format of a <DaylightInfo> argument is as follows (in one single line):

```
<StdShift>,<DltShift>,<SDay>
,<SMonth>,<SDayPos>,<SHour>
,<DDay>,<DMonth>,<DDayPos>,<DHour>
```

This argument is made up of several sub-arguments as shown below:

Sub-argument	Description	Possible values
<StdShift>	Expressed in minutes, this describes the time difference between standard time within a time zone and the time of the time zone concerned. For example, for Paris (GMT+1 time zone), if <StdShift> is set to 30 (minutes), standard time within this time zone is GMT+1h30min and not GMT+1h.	By default, this sub-argument is null, but it can be set to any numeric value. The user must verify the coherency of this sub-argument.

Sub-argument	Description	Possible values
<DltShift>	Expressed in minutes, this describes the time difference between daylight saving time and the time of the time zone concerned.	By default, this sub-argument is set to 60 (which corresponds to 1 hour's time difference between daylight saving time and the reference time "GMT +") but it can be set to any numeric value. The user must verify the coherency of this sub-argument.
<SDay>	Day of change from daylight saving time to standard time.	"Monday" "Tuesday" "Wednesday" "Thursday" "Friday" "Saturday" "Sunday" Empty (in this case, you need to set <SDayPos> to a value between 1 and 31 to identify the day of change from standard time to daylight saving time)
<SMonth>	Month of change from daylight saving time to standard time.	"January" "February" "March" "April" etc. "November" "December"
<SDayPos>	Position in the month of the day of change from daylight saving time to standard time.	"First" "Second" "Third" "Fourth" "Last" "Penultimate" (before-last) A value between 1 and 31 when <SDay> is empty
<SHour>	Time of change from daylight saving time to standard time (expressed in daylight saving time).	Any value expressed in 24 hour format (HH:MM:SS).

Sub-argument	Description	Possible values
<DDay>	Day of change from standard time to daylight saving time.	"Monday" "Tuesday" "Wednesday" "Thursday" "Friday" "Saturday" "Sunday" Empty (in this case, you need to set <DDayPos> to a value between 1 and 31 to identify the date of change from daylight saving time to standard time.)
<DMonth>	Month of change from standard time to daylight saving time.	"January" "February" "March" "April" etc. "November" "December"
<DDay-Pos>	Position in the month of the day of change from standard time to daylight saving time.	"First" "Second" "Third" "Fourth" "Last" "Penultimate" (before last) A value between 1 and 31 when <DDay> is empty
<DHour>	Time of change from standard time to daylight saving time (expressed in standard time).	Any value expressed in 24 hour format (HH:MM:SS).

Example

As an example, lets take the information for daylight saving time for "(GMT+01:00) Paris, Madrid, Amsterdam" time zone.

```
2000=0,60,Sunday,October,last,03:00:00,Sunday,March,last,02:00:00|0,60,Su
nday,September,last,03:00:00,Sunday,March,last,02:00:00
```

Let's identify the parameters used. The parameters are separated by the pipe character "|". In this case, two parameters are used:

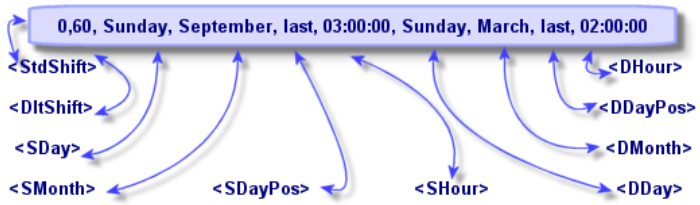
```
2000=0,60,Sunday,October,last,03:00:00,Sunday,March,last,02:00:00
0,60,Sunday,September,last,03:00:00,Sunday,March,last,02:00:00
```

First parameter

- <Year> = 2000 means that the following <DaylightInfo> parameters are applicable from 2000 onwards.
- <StdShift> = 0 means that there is no difference between the time zone and standard time within this time zone.
- <DltShift> = 60 means that the time difference between standard time and daylight saving time is 60 minutes, i.e. 1 hour. Daylight saving time is therefore equal to the time of the time zone plus one hour.
- <SDay> = Sunday means that the change to standard time takes place on a Sunday.
- <SMonth> = October means that the change to standard time takes place during the month of October.
- <SDayPos> = Last specifies the position of the day in the month. Here the change to standard time takes place on the last Sunday of the month of October.
- <SHour> = 03:00:00 means that the change to standard time takes place at 3 A.M.
- <DDay> = Sunday means that the change to daylight time takes place on a Sunday.
- <DMonth> = March means that the change to daylight time takes place in the month of March.

- <DDayPos> = Last specifies the position of the day within the month. In this case, the change to daylight time takes place on the last Sunday of the month of March.
- <DHour> = 02:00:00 means that the change to daylight takes place at 2 A.M.

Second parameter



- Since there is no <Year> argument, this parameter specifies that it is only valid for those years not described in the previous parameter.
- <StdShift> = 0 means that there is no difference between the time zone and standard time within this time zone. Winter time is thus equal to the time zone time.
- <DltShift> = 60 means that the time difference between standard time and daylight saving time is 60 minutes, i.e. 1 hour. Daylight saving time is therefore equal to the time of the time zone plus one hour.
- <SDay> = Sunday means that the change to standard time takes place on a Sunday.
- <SMonth> = September means that the change to standard time takes place during the month of September.
- <SDayPos> = Last specifies the position of the day in the month. Here the change to standard time takes place on the last Sunday of the month of September.
- <SHour> = 03:00:00 means that the change to standard time takes place at 3 A.M.
- <DDay> = Sunday means that the change to daylight time takes place on a Sunday.
- <DMonth> = March means that the change to daylight time takes place in the month of March.

- <DDayPos> = Last specifies the position of the day within the month. In this case, the change to daylight time takes place on the last Sunday of the month of March.
- <DHour> = 02:00:00 means that the change to daylight takes place at 2 A.M.

As a result:

Note:

From 2000 onwards, the change to standard time is made on the last Sunday of October at 03:00:00 (clocks go backward to 02:00:00) and the change to daylight saving time takes place on the last Sunday of March at 02:00:00 (clocks go forward to 03:00:00).

For all years before 2000, the change to standard time is made on the last Sunday of September at 03:00:00 and the change to daylight saving time takes place on the last Sunday of March at 02:00:00.

Managing time zones in AssetCenter Server

AssetCenter Server enables you to configure tests concerning time zones. Select the **Tools/ Configure modules** menu item.

Tests to perform

In the **General** tab of the configuration screen, you configure the types of text to be performed:

- Verify time zone of database server.
- Verify local time compared to that of the server.

These two tests compare the time of the database server with that of the machine on which AssetCenter Server is installed. The time difference is expressed as [(n * 30minutes) + m] where m is between -15 minutes and + 15 minutes.

In both cases

If the time difference exceeds 5 minutes, AssetCenter Server offers to update the local time on the machine on which it is installed.

If you refuse this update (for example, if you think that if is the time of the server that requires changing), the connection is refused. You can connect again one the difference between the two times no longer exceeds 5 minutes (resulting from either a modification to the time of the database server and/or of the machine on which AssetCenter Server, is installed).

Specific aspects of the Verify time zone of database server option

Note:

To do this, the machine on which AssetCenter Server is running must have the correct time and have the correct information on daylight saving changes.

If necessary, the information on the time zone of the server in the table of options of AssetCenter is updated (if the number (n * 30 minutes) does not correspond to the time zone of the server).

Specific aspects of the Verify local time compared to that of the server option

The time zone of the server, necessary for internal operations of AssetCenter, is recovered.

Frequency of the test

The test is performed:

1. First, when AssetCenter Server connects to the database.
2. Then regularly, according to the schedule you specify in AssetCenter Server (**Tools/Configure modules**).

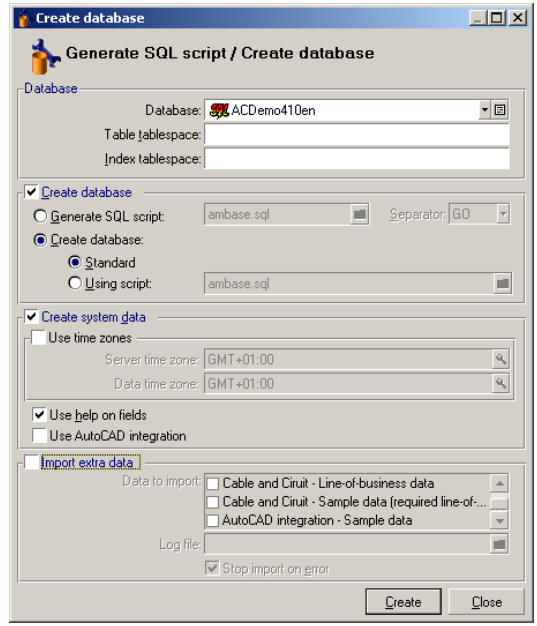
Consequences for various operations

Time zones have an impact on a number of operations:

- Creating the database
- Connecting to a database
- Import and Export

Creating the database

When creating the database, AssetCenter allows you to define time zone options. Select the **Action/ Create database** menu item in AssetCenter Database Administrator. The **Create system data** frame contains the options concerning time zones.



The **Use time zones** check box determines whether time zones are taken into account when creating the database.

- If the box is checked, time zones are used when creating the database.
- If this check box is cleared, time zones are ignored when creating the database.

The **Server time zone** and **Data time zone** determine the effective time zone of the server at the moment of creation of the database and the time zone according to which the data will be stored.

Note:

This option is only accessible when creating a database. It allows you to define the time reference of the server and of the data. If you change these values, the "Date time" values already contained in the database do not make sense.

Connecting to a database

When connecting to a database, AssetCenter searches in **aam.ini**, (situated in the root folder of windows) for the "LocalTimeZone" entry that defines the time zone of the client machine.

If this information cannot be found, AssetCenter uses the system time zone (defined under Windows NT or Windows 95).

AssetCenter next searches the database for the time zone corresponding to the "LocalTimeZone" entry in **aam.ini**, or the time zone defined in Windows.

The table below summarizes the different possible situations:

"LocalTimeZone" exists in aam.ini?	Corresponding time zone found in table of time zones?	Information saved in aam.ini file for the "LocalTimeZone" entry
Yes	Yes	Database time zone
	No	Unchanged
No	Yes	Database time zone
	No	System time zone

Adjusting time on client machine

On connecting to a remote database, AssetCenter verifies the validity of the time given by the clock at client level with reference to the server clock.

The clock differential is a difference in synchronization and is not to be confused with the time difference, which is a transposition in relation to a difference in time zones. AssetCenter calculates the time zone of the client clock and determines the clock differential between the two machines. This calculation is made as follows:

```
Differential = Modulo((Difference in minutes between the times of the two
 machines in question)/30)
```

Note:

The modulus is the remainder of a division.

For example, for the following machines:

- Machine A is in GMT and is showing 18:02
- Machine B is in GMT+1 and is showing 18:19 (i.e. 17:19 for machine A, 43 minutes ' difference with machine A))

```
Differential = Modulo (43/30)= 13 minutes
```

If this difference exceeds five minutes (fixed value), AssetCenter proposes adjusting clock time at the client level.

Connection fails if the user refuses.

AssetCenter performs this check periodically and when time is changed on the client machine. By default, this check is made every 60 minutes, but this can be configured by modifying the **g_lTimeZoneCheckInMns** option in **aam.ini** (situated in the root folder of Windows), in the [option] section.

```
[option]
g_lTimeZoneCheckInMns = 30
```

The frequency of the clock differential check is set to 30 minutes.

This frequency can also be configured via the **Verify database server time zone** option in AssetCenter.

Note:

This verification only functions if the database concerned has been created with time zone taken into account.

Import and Export

For these two functions, the conversion is made assuming that all "Date and time" fields correspond to the time zone of the machine carrying out the import or export.

13 Calculated fields

CHAPTER

You access the screen to create calculated field via the **Administration/ Calculated fields** menu.

Definition of a calculated field

A calculated field is a field whose value is calculated according to the value of other fields and variables, using a user-defined formula. There are three types of calculated field:

- AQL
- Basic
- Calculated field

Each of these types uses a different language for the calculation formulas and has an effect on the possibilities and constraints linked to using the field. For example, only "AQL" type calculated fields can be used in filters.

Note:

Calculated fields are virtual fields that are read-only (the formula alone is stored in the database). You can define as many calculated fields as you like and assign user rights to them.

Usefulness of calculated fields

Calculated fields make it possible to define additional information and to calculate synthetic information concerning records of a table in the AssetCenter database. In this way, apart from a few differences, they resemble "classic"' database fields:

- Unlike "classic" fields, the value of a calculated field is not stored in the AssetCenter database.
- The value of a calculated field is not populated by a user but given by a formula.
- You cannot associate a calculated field with a single record of a given field. Just like all other "classic" fields in the database, a calculated field is associated with all the records of a table and has a value (can be null) for each record in this table.
- Calculated fields do not appear in the detail screen of a record. They can only be displayed in a list.
- Calculated fields can only be used in the calculation of the default values of standard fields if their type is **Calculated string** or **Basic script**.

Creating a calculated field

Before creating such a field, it is useful to be acquainted with the specific details inherent to each type of calculated field.

This section includes information on the following topics:

- Introduction
- Methods of creation

Introduction

Each type of calculated field has different properties that determine how it is used.

The following table resumes the main differences between the three types:

Table 13.1. Calculated fields types

Type of field	Properties of fields of this type			Field calculated by	Characteristics of the language used by the calculation formula	
	Can be displayed	Can be sorted	Can be used in filters		Advantages	Disadvantages
AQL	Yes	Yes	Yes	The database server	Powerful Integrated editor	Limited language. Fields of this type cannot be used in default values.
Calculated strings	Yes	Yes	No	The client	Simple	Not very powerful (simple concatenation of strings and values of fields and strings only).
Basic	Yes	No	No	The client	Many possibilities Flexible	Fields of this type can be displayed only.

Note:

This table clearly shows that "AQL" type fields have much wider range of application than the two other types of calculated fields.

An AQL query can make use of all three properties (can be displayed, sorted, used in filters):

Property	Corresponding AQL arguments
Can be displayed	`SELECT` clause
Can be sorted	`SELECT` `ORDER BY` `GROUP BY` clauses
Can be used in filters	`SELECT` `ORDER BY` `GROUP BY` `WHERE` `HAVING` clauses

For further information on AQL queries, refer to the manual entitled "Reference Guide: Administration and Advanced Use" chapter "Writing queries in AQL".

Calculation by the server / client machine

In the case of an "AQL" type field, the database server performs the necessary calculations and sends the result to the client machine. For this reason, there is no impact on the speed of the client machine and network traffic is reduced. On the other hand, the SQL queries sent to the database are more complex.

Methods of creation

This chapter describes in detail the method used to create a calculated field.

Analyze your needs

Choose a type of field based on the two following criteria:

- The properties of the type of field: Can be displayed, sorted, used in filters or default values.
- The "cost" of this solution; in terms of complexity of the formula used compared to the possibilities of utilization. The three types of calculated field can be classified as follows (in terms of increasing cost:
  - Calculated field
  - AQL
  - Basic

Note:

Whenever possible, we recommend using the least "costly" solution.

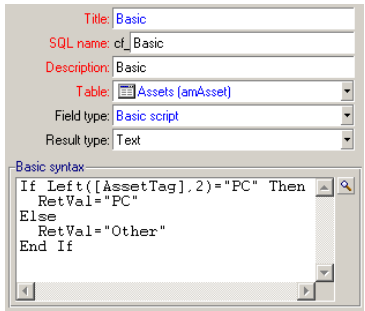
For example:

- If the field is for informational purposes only, a Basic type calculated field will suffice.
- If you want to be able to sort records according to the value of the field, "AQL" or "Calculated string" must be used.
- If you want to be able to filter records according to the value of the field, "AQL" must be used.

Once you have defined your needs, you can move on to the next phase.

Open the creation screen

Select the **Administration/ Calculated fields** menu item. AssetCenter displays the calculated field creation screen:



Identify the calculated field

First, populate the upper part of this screen in order to uniquely identify the calculated field:

- The **Title** field (SQL name: Label) contains the label of the calculated field, used for column headers in lists.
- The **SQL name** field (SQL name: SQLName) contains the SQL name of the calculated field. This name, prefixed by the letters "cf_", is used, for example, when referring to this field in Basic scripts, queries or filters.

  Note:

  You must not modify the SQL name of a field once it has been created. Any references to this field using the previous SQL name will no longer be valid.

- The **Description** field (SQL name: Description) contains a short description of the field, used in the lists showing the fields (in filters or the list configuration screen, for example).

Define the context of utilization of the field

The **Table** (SQL name: TableName) and **Field type** (SQL name: seType) fields allow you to define the context of utilization of a calculated field:

- The **Table** field (SQL name: TableName) allows you to associate a calculated field with a table. The field will only be available for this table.
- The **Field type** field (SQL name: seType) allows you to specify the type of the calculated field. According to this type, the properties of the field (can be displayed, sorted or used in filters) will differ.
- The **Result type** field allows you to specify the resulting type of the calculated field. This type is used for formatting and display purposes. A calculated field whose result type is a date is thus displayed in the same way as all other "Date" type fields in the database.

Enter the calculation formula for the field

Now you just have to write the field calculation formula. You can either enter this directly in the text field in the lower part of the screen (note that the label of this field changes according to the attributed type), or click the magnifier or press "F4" to access the corresponding editor screen.

Note:

The language used differs according to the type of field used.

For further information on the languages available for writing a calculation formula, please refer to the following documentation:

- Manual entitled "Reference manual: Administration and advanced use", chapter "Using scripts" for the Basic language. The function used is **RetVal()**.
- Manual entitled "Reference manual: Administration and advanced use", chapter "Writing queries in AQL" for the AQL language.
- Manual entitled "Reference manual: Administration and advanced use", chapter "Structure of the AssetCenter database", paragraph "Description of the tables", sub-paragraph "AssetCenter table description strings" for calculated strings.

Define the user rights for the calculated field

Select the **Administration/ User rights** menu item. AssetCenter displays the screen for creating user rights.

Note:

Calculated fields are accessible for status access only.

1. Enter a brief description for the user right in the **Description** (SQL name: Description) field (SQL name: Description) and, optionally, a comment in the **Comment** field (SQL name: Comment).
2. Expand the tree structure of the table associated with the calculated field. The branch identified by the f(x) (Calculated fields) icon includes a full list of the calculated fields for the table in question.
3. Then select the field for which you want to edit a user right. The **Read** check box in the **Fields, links and features** zone allows you to define the read rights for this field. When this box is checked, only profiles with this user right can view the calculated field. If the box is not checked, then all users will have (read-only) access to this field.

Using calculated fields

How a calculated field can be used depends on its type. You need to make sure that the type is compatible with its intended utilization. In lists showing fields (creating a filter, configuring a list, etc.), AssetCenter helps you by only showing those fields that can be used.

Using a calculated field in the configuration of a list

You can display the value of a field calculated for all records in a table using the **Configure list** command in the shortcut menu.

In the list of fields and links of a table, unfold the **Calculated fields** branch to access the available calculated fields. AssetCenter displays the calculated fields in the defined format via the **Edit/Options** menu, **Display** category.

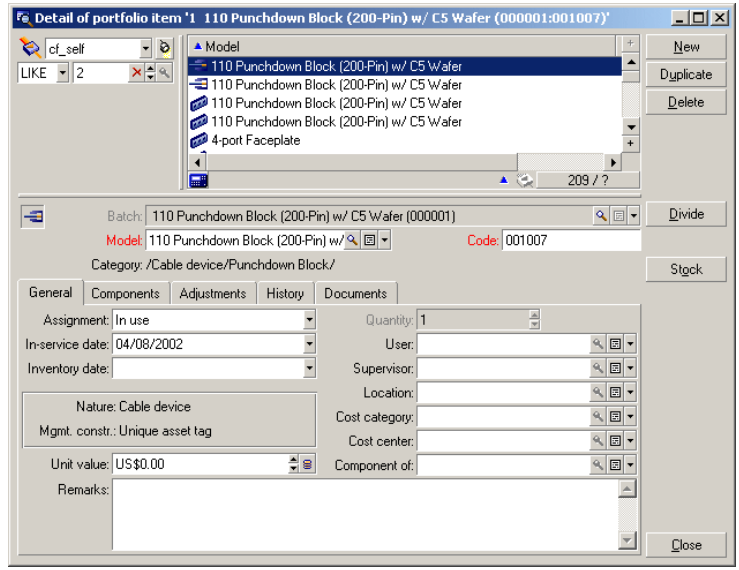
You can add these to the list in the same way as for any other field.

Filtering the records of a table

AssetCenter can filter the records of a table according to the value of an "AQL" type calculated field. To do this, select the **Simple filter** command from the shortcut menu and go to the **Calculated fields** branch. AssetCenter only shows the "AQL" type fields.

Referencing a calculated field

The SQL name, prefixed by the letters "cf_" is used to reference calculated fields. The screen copy below illustrates the use of the SQL name of a calculated field in a filter:



Additionally, calculated fields can be used by the different modules or functions of AssetCenter:

- AssetCenter Web
- AssetCenter APIs
- Reports
- Forms

For further information on these modules or areas of functionality, please refer to the corresponding documentation.

CHAPTER 14 Wizards

AssetCenter lets you create your own wizards and adapt existing wizard for your own use. Wizards are stored as text fields (**Wizard script** field (SQL name: WizardScript), **Wizard** tab of **Assistant** type action detail). Creating a wizard consists of entering its code in this field directly or using the graphical editor. Doing this requires familiarity with the structure of wizards and the scripting language used to describe this structure.

Notational conventions

The following notation is used to describe the structure of wizards:

Table 14.1. Conventions used

[ ]	Square brackets are used to reference the value of a field in the database (in the case of contextual wizards) or one of the "special fields": "CurrentSelection" and "CurrentTable". They are also used to indicate optional parameters.

< >	Angle brackets are denote values for properties described in plain language. These angle brackets are not meant to be typed as is. Replace them and the text within with the appropriate information.
|	The pipe character is used to separate possible values for a property. It is also used to separate the titles and values of multi-column lists.
{ }	Curly bracket frame the definition of a node or a block of script spanning several lines for a property. They are also used to reference the value of a wizard property.
'	In examples of Basic code, the apostrophe designates a line of comments that is not interpreted by AssetCenter.
; or //	In the wizards, the semicolon or two slashes designate a line of comments, which is not interpreted by AssetCenter.

Definitions

You will find below definitions of the terms used in the description of the structure of wizards.

Twip

The "Twip" is unit of size, and distance by default, used by the wizards. It is independent of the resolution of the screen. It corresponds to:

- 1440 "twips" equals an inch.
- 567 "twips" equals a centimeter.
- In 96 dpi resolution (standard Windows resolution) 15 twips is equivalent to 1 pixel.

Control

A control designates a graphical element that allows you to edit a data item. Common controls are check boxes, text edits, buttons, drop-down lists, etc.

Node

A node corresponds to a hierarchical level in the tree-structure of the wizard. A sub-node of a given node "N", is a node one level down in the tree, attached to node "N".

Note:

Only accented characters are not allowed in the names of nodes. The names of nodes are limited to 22 characters.

Object

An object is a generic term that designates:

- An entire wizard.
- A wizard page.
- A control (check box, text edit, button, drop-down list, etc.) in a page.
- A variable.
- Etc.

Parent object and child object

If an object "A" contains an object "B":

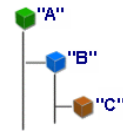
- Object "A" is called the parent object of object "B".
- Object "B" is called the "child object" of object "A".

Warning:

This relationship concerns composition and not inheritance.

Full name of an object

The full name of an object is made up of the name of all its parent objects and the name of the object itself. Each object is separated by a period ("."). Let's take the following structure as an example:



The full name of object "C" is therefore:

```
<Name of the object "A">.<Name of the object "B">.<Name of the object "C"
>
```

Variable

A variable is a named storage space containing data that can be modified during the execution of a wizard. Each variable has a name allowing it to be identified uniquely within the wizard. All variables used in the wizard are global. This means that they can be referenced in any given node of the wizard using their full name.

Wizards in AssetCenter use two types of variable:

- "Wizard variables" that are defined in "LONG" or "STRING" type nodes. The type of the node defines the type of the variable; A variable defined in a "LONG" node is a long integer, a variable defined in a "STRING" node is a character string. These variables are, by definition, global. That is to say that they can be referenced using their full name from any node in the wizard. If necessary, these variables are recalculated automatically by AssetCenter.
- Basic variables, used in Basic scripts within the wizard. By default, these variables are local, but can be made global using the "COMMON" and "GLOBAL" properties. These variables are not recalculated automatically by AssetCenter.

Transition

A transition designates what happens from one page in the wizard to another. Several transitions can be defined for a given page. Each one of these has its own user-defined conditions that determine its validity and must by fulfilled in order to trigger the transition:

- When the user clicks **Next**, the first valid transition is executed (i.e. that for which the conditions are met). If no transitions are valid, the **Next** button is disabled.
- If the wizard has mandatory properties that are not populated, you cannot use the **Next** button.
- If the user clicks **Finish** before going through each step of the wizard in full, the default values for the unfinished steps are used.

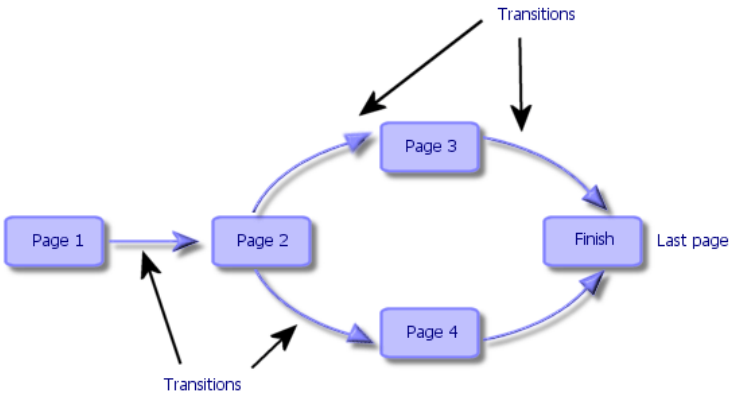
Structural template

An AssetCenter wizard results from the execution of a wizard. By definition, the structure of the wizard is based on the structure script. I.e.:

- A wizard script (and therefore a wizard) is made up of nodes.
- Each node of the wizard has a name, one or more sub-nodes and a set of properties. The node types are listed below:
  - "ROOT" (Root node). This node is unique and encompasses all the others.
  - "START". This node is unique and contains a script that is executed when the wizard is started up.
  - "PAGE". This type of node describes a wizard page.
  - "TRANSITION". This type of node describes the transition between two "PAGE" type nodes.
  - "FINISH". This node is unique and contains a script that is executed at the end of the wizard.
  - "PARAMS". This node is unique and contains the parameters to be passed to another wizard. Several wizards can be executed in succession (with or without exchange of parameters). These wizards are said to be chained.
  - "LONG" or "STRING". This type of node defines a corresponding variable type.
- The value of a property is made explicit either via a constant or via a Basic script (in this case, the value results from the evaluation of the script).

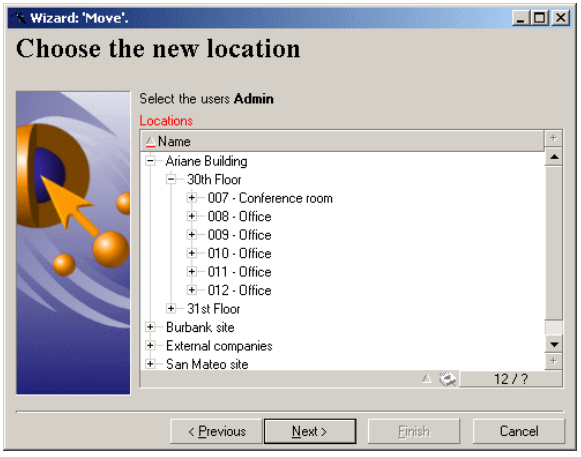
Wizards are made up of pages linked by transitions. The branching from one page to another is determined by the information entered the choices made by the user. The following diagram illustrates the structure of a wizard:

Figure 14.1. Wizards - structural templates



Wizard page template

A wizard page is organized as follows:



General points

The code of a wizard (**Wizard script** field (SQL name: WizardScript) in the **Wizard** tab of the detail of an action using a wizard) takes the forms of structured text, constituted of blocks delimited by braces ({ }). This text defines the structure of a wizard.

Each node ("Root", "Page", etc.) in the tree of the wizard has an unlimited number of sub-nodes and a set of properties.

Generic structure and syntax

Nodes have the following structure and syntax:

```
; This is a comment outside of the script
{ <Node type> <Node name>
 <Name of the property> = <Value of the property>
 ' This is a comment inside the script
 ...
 { <Name of the property> =
  ...
 }
 { <Node type> <Node name>
  <Name of the property> = <Value of the property>
  ...
  { <Name of the property> =
   ...
  }
 }
}
```

The following rules are applied to nodes:

- The names of nodes are optional. If no name is specified for a node, AssetCenter assigns a name and number to it automatically.
- The names of nodes may not include spaces.
- If the name of a node is "=", it is no longer a node but a multi-line property. For further information on multi-line properties, please refer to "Syntax of properties" in this chapter.
- Lines beginning with a semicolon (";") outside of a script and lines beginning with an apostrophe (') within a script are interpreted as comments and are ignored.

Note:

Attention: The space between the ("{") brace and the node type must be respected. If this is not the case, AssetCenter will refuse to execute the wizard.

Properties of a node

The values of properties can be defined using constants or scripts. Constants can be numeric, Boolean, or text.

Note:

The properties associated with objects can be optional or mandatory. They can either be "logical" (they complement the definition of the object) or "physical" (they have an impact on the visual aspect of the object).

Declarative template

A property is defined according to a declarative mode that defines circular references (A={B}, B={A}):

```
<Name of the property>=<Script>
```

A list of dependencies is associated with this definition. If we have:

```
A={B}+{C}
```

Property "A" depends on properties "B" and "C". This list of dependencies of "A" is therefore: "B", "C".

As a result, a property changes:

- If one of the properties in its list of dependencies changes.
- Following a user action that changes a property or a dependent property.

Defining a constant as a value for a property

The following syntaxes define a constant value for a property:

- Text type property:
  - <Name of the property> = "<Text>"

- Boolean type property:
  - <Name of the property> = TRUE
  - <Name of the property> = FALSE
  - <Name of the property> (equivalent to <Name of the property> = TRUE)
- Numeric type property:
  - <Name of the property> = 42
- <Name of the property> = {<Full name of a Basic variable or a property>}

Note:

The Boolean value "TRUE" is equivalent to a numeric value other than "O". "FALSE" is equivalent to the numeric value "0".

Referencing a property

To reference a property of an object (I.e. make reference to the contents of this property or object, and in particular its value), the syntax is as follows:

```
{<Full name of the property>}
```

Thus, if you want to reference the property "Prop" of a page "Page1", you write:

```
{Page1.Prop}
```

In this syntax, the full name is case insensitive.

Defining a script as value for a property

Notion of script

A script is a single or multi-line Basic program that returns a value in the global variable "RetVal". In the case of a single line script, this variable is implicit. In the case of a multiple-line script you must specify it.

As is the case for all Basic scripts, pay attention to the type of the returned value. This depends on the type of the property calculated via the script.

Syntax of a single line script

```
<Name of the property>=<Script>
```

For example:

```
Variable="The name is: " & {Name}
```

The previous single line script is equivalent to the following multiple line script:

```
{ Variable =
RetVal="The name is: " & {Name}
}
```

Syntax of a multiple-line script

```
{ <Name of the property >=
 <Script>
}
```

For example:

```
{ LABEL =
 IF {Page1.Title}="Choose an employee" THEN
RetVal="Employee"
 ELSE
RetVal="Department"
 END IF
}
```

Methods applicable in properties

A method allows you to recover a value linked to a property or node or even execute a function on this property. In this sense, it can be considered as an advanced function.

The syntax of a method is as follows:

```
{node.node.node[.property][.method([arg1[, arg2[ ]]])]}
```

with:

- node: name of the node
- property: name of the property
- method: name of the method
- arg1, arg2, etc.: Basic statement or expression (this must not contain bracket characters).

Note:

In this example, the characters "[" and "]" frame optional items.

For example, to recover the number of lines from the "LISTBOX" control in page "PAGE1", we use the "COUNT" method associated with this type of control. The command is as follows:

```
{PAGE1.LISTBOX.VALUES.COUNT()}
```

Table type property

Table type properties are properties whose value is defined according to the following format:

```
<Columm|Columm|Columm|...>=<Id of the line>, <Columm|Columm|Columm|...>=<
Id of the line>, ...
```

The values of these properties can be viewed in the form of a table:

		Column 1	Column 2	Column 3
Line number: 1	**Id of the line (e.g.: 18)**	Cell (1,1)	Cell (2,1)	Cell (3,1)
Line number: 2	**Id of the line (e.g.: 29)**	Cell (1,2)	Cell (2,2)	Cell (3,2)
Line number: 3	**Id of the line (e.g.: 78)**	Cell (1,3)	Cell (2,3)	Cell (3,3)
Etc.	Etc.	Etc.	Etc.	Etc.

Note:

The identifier is a 'Text'-type identifier.

Example

Let's consider the "VALUES" property of the "LISTBOX" node having as its value the result of a query on the Departments and employees table. The query in question returns the values of the **Name** (SQL name: Name) and **First** (SQL name: FirstName) fields for each record in this table. Let's suppose that this property has the following value:

```
VALUES="Colombo|Gérard=32,Lubeck|Alexandre=64,Daquin|William=24"
```

This value can be viewed in the form of a table:

		Name	First name
1	**32**	Colombo	Gerard
2	**64**	Lübeck	Alexander
3	**24**	Daquin	William

Using the global variables CurrentTable and CurrentSelection

The contents of these variables can be recovered using the following syntax:

```
[CurrentTable]
[CurrentSelection]
```

The following table presents the features of these two variables:

Name of the variable	Description of the variable	Comment
CurrentTable	Contains the SQL name of the active table at the time the wizard is launched. If there is no active table, it contains an empty string. "String" type variable.	This variable is automatically populated by AssetCenter. The user cannot force the value.
CurrentSelection	Contains the list of internal identifiers of records selected at the time the wizard is launched, separated by a comma. "String" type variable.	This variable is automatically populated by AssetCenter>. It contains an empty string if there is no selection or if no tables have been updated. The user cannot force the value.

Sequencing wizards

Once executed, a wizard can be used to trigger the execution of another wizard and pass parameters (variables) to this wizard. This is known as sequencing wizards.

Execution

In order for a wizard A to trigger a wizard B, its Finish not must have the CHAIN property. This property must have the value of the SQL name of the **Assistant** type action to be executed, in this case "B".

Parameters

Parameters are passed to wizard B using the PARAMS node of wizard A. These parameters are added to those of the PARAMS node of wizard B. If same parameter is defined for both the PARAMS node of wizard A and in the PARAMS node of wizard B, the parameter in wizard takes precedence over that in wizard B.

Basic functions

In addition to the generic functions of AssetCenter (with the exception of the "AmCounter" function), wizards accept the following additional functions:

- AmComputeString()
- AmDecrementLogLevel()
- AmExecTransition()
- AmLog()
- AmMsgBox()
- AmPagePath()
- AmProgress()
- AmRefreshProperties
- AmSetProperty
- AmUpdateDetail
- AmValueOf
- AmWizChain

Warning:

When you call Basic functions in a wizard script, you must always assign the value returned by the function to a variable. Otherwise the Basic compiler will return an erro. The following example will not compile:

```
AmGetFieldLongValue(hRecord, "lUserId", {lEmplDeptId})
```

The correct script is as follows:

```
Dim lValue as Long
lValue=AmGetFieldLongValue(hRecord, "lUserId", {lEmplDeptId})
```

Definition of a Root node

The "Root" node describes the wizard as a whole. It is made up of a block of general properties, which can be applied to all the wizards and series of sub-nodes that represent objects contained in the wizard.

Syntax of a Root node

The syntax of a "Root" node is as follows:

```
' Block of general properties of the root node
NAME=...
IMAGE=...
...
' Definition of sub-nodes of the root node
{ FINISH
  ...
}
{ PAGE
  ...
}
{ TRANSITION
  ...
}
```

Properties of a Root node

The following tables list all the logical and physical properties that can be defined in a "Root" node:

Table 14.2. Logical properties of the "Root" node

Name of the property=Value	Description of the property	Example	Comment
NAME="<Name of the wizard>"	Defines the title of the window of the wizard. "String" type property.	NAME = "MoveWizard"	You must define a value for this property. The name of the wizard is limited to 22 characters. This property is used to for the serialization of the wizard: The data relating to this wizard is stored under this name. As a consequence, it is preferable for different wizards to have different names.
TITLE=<"Title of the window>"	Defines the name of the wizard. "String" type property.	TITLE=" Move wizard"	We strongly recommend defining a value for this property.
GLOBAL=<Script>	Allows you to serialize the wizard (=TRUE) or not (=FALSE). If the wizard is serialized, it conserves (storing them in the "ini" file) the values previously entered for the next time it is executed. This script declares, but is not executed. To execute a script at startup, use the "START" node. "Boolean" type property.	{GLOBAL=Dim Filter As String}	

Name of the property=Value	Description of the property	Example	Comment
COMMON	Contains the Basic functions automatically included in all wizards. This property corresponds to the **gbbase.wiz** file (read only), which is inserted into the database when it is created.		
SERIALIZE=<TRUE|FALSE>	Allows you to serialize the wizard (=TRUE) or not (=FALSE). If the wizard is serialized, it conserves (storing them in the "ini" file) the values previously entered for the next time it is executed. The NAME property determines in which section of the **.ini** file the values will be stored. "Boolean" type property.	SERIALIZE=TRUE	By default, this property is set to "FALSE".
MODAL=<TRUE|FALSE>	Defines whether the wizard is modal (=TRUE) or not (=FALSE).		

Table 14.3. Physical properties of the "Root" node

Name of the property=Value	Description of the property	Example	Comment
IMAGE="<Path of the bitmap file> "IMAGE16="<Path of the bitmap file>"	Defines the bitmap type graphical file (.bmp) to be displayed in the wizard. "String" type property.	IMAGE="C:\Images\Page1.bmp"	If no value is defined for this property, then no image will be displayed. The path of the graphics file depends on the AssetCenter Config folder. AssetCenter first searches for the imagine in the database If you define a value for "IMAGE16", this property is used instead of "IMAGE" when the color depth of the screen is 16.
WIDTH=<Width>	Defines the default width ("<Width>") of the window of the wizard. This is expressed in twips. "Long" type property.	WIDTH=6000	

Name of the property=Value	Description of the property	Example	Comment
HEIGHT=<Height>	Defines the default height ("<Height>") of the window of the wizard. This is expressed in twips. "Long" type property.	HEIGHT=5000	
MINWIDTH=<MinWidth>	Defines the minimum width of the wizard window. Value expressed in **twips**.		
MINHEIGHT=<MinHeight>	Defines the maximum height of the wizard window. Value expressed in **twips**.		
CTRLHEIGHT=<CtrlHeight>	Defines the height of a control whose vertical size is set (for example, a TEXTBOX control). Spacing value expressed in **twips**.		
LABELSPACING=<labelSpacing>	Defines the space between the label of a control and the control itself (when the title is above the control). Spacing value expressed in **twips**.		

Name of the property=Value	Description of the property	Example	Comment
CTRLSPACING=<CtrlSpacing>	Defines the vertical spacing between two controls. Spacing value expressed in **twips**.		
IMGBORDER=<Width>	Defines the horizontal spacing between the wizard image and its controls. Value expressed in **twips**.		
NAVIGATION=<TRUE|FALSE>	Displays (=TRUE) or not (=FALSE) that navigation bar containing the **Next** and **Cancel** buttons in the wizard window.		
CONFIRMCANCEL=<TRUE|FALSE>	Displays (=TRUE) or not (=FALSE) the message confirming the cancellation.		
DEFAULTON-NEXT=<TRUE|FALSE>	Selects by default (=TRUE) the **Next** button. If DEFAULTON-NEXT=FALSE then the button selected by default is **End**.		

Sub-nodes of a Root node

The types of sub-nodes that you can define for a root node are listed in the table below. Each type of node represents an "Object".

Table 14.4. Sub-nodes of "Root" node

Node type	Description
PAGE	Describes a page of the wizard.
FINISH	Describes the final transition from the final page of the wizard (to the finish). This "Transition" type node does not have "FROM" and "TO" properties.
START	Contains, for example, a script to be executed when the wizard is launched (using the "DO" property) and the name of the starting page of the wizard ("TO" property).
PARAMS	Enables you to transfer parameters from one wizard to another (if the CHAIN property of the FINISH sub-node is populated).
TIMER	Enables you to associate a timer with a wizard page.

Definition of a Page node

A "Page" node describes a page in a wizard. It is made up of a block of properties applicable to this node and all its sub-nodes; and a set of sub-nodes that define objects defined in the page.

Syntax of a Page node

The syntax of a "Page" node is as follows:

```
' Declaration of the page
{ Page <Name of the page>
' Block properties of the page node
IMAGE=...
TITLE=...
```

```
' Definition of sub-nodes of the "Page" node
{ TRANSITION
  ...
}
{ <Control type> <Control name>
  ...
}
  ...
}
```

Properties of a Page node

The following tables list all the properties that can be defined in a "Page" node:

Table 14.5. Logical properties of a "Page" node

Name of the property=Value	Description of the property	Example	Comment
TITLE="<Title of the page>"	Defines the title of the page. This title appears in bold at the top of the page. "String" type property.	TITLE="Move"	If no value is defined for this property, it inherits the value of the "TITLE" property of the "Root" node. Unlike labels, this string does not support HTML.
ONENTER=<Script>	Defines a Basic script that is executed when the page is accessed by clicking **Next** or **Previous**. "Boolean" type property.	{ONENTER = AmMsgBox ("Hello") }	

Table 14.6. Physical properties of a "Page" node

Name of the property=Value	Description of the property	Example	Comment
IMAGE="<Path of the bitmap file>" IMAGE16="<Path of the bitmap file>"	Defines the bitmap type graphical file (.bmp) to be displayed in the page of the wizard. "Boolean" type property.	IMAGE =" C:\Images\Page1.bmp"	If no value is defined for this property, it inherits the value of the "IMAGE" property of the "Root" node. If you define an empty value for this property, no image is displayed. If you define a value for "IMAGE16", this property is used instead of "IMAGE" when the color depth of the screen is 16.

Sub-nodes of a Page node

Two types of sub-nodes can be defined for a "Page" node:

Table 14.7. Sub-nodes of "Page" node

Node type / "Object"	Description
<Control type> <Control name>	Defines a control displayed in the current page.
TRANSITION <Name of the transition>	Describes a transition between the current page and another page of the wizard
TIMER	Enables you to associate a timer with a wizard page.

Definition of a Transition node

A "Transition" node describes the passage between two pages in a wizard. It is exclusively made up of a block of properties.

Note:

Transitions can be defined from the inside of a "Page" node (in this case, they do not require the "FROM" property) or in the "Root" node. The final transition leading to the closure of the wizard, is described in a "FINISH" node (at "Root" node level) and does not have "FROM" and "TO" properties.

Syntax of a "Transition" node

The syntax of a "Transition" node is as follows:

```
' Declaration of the transition
{ TRANSITION0 <Name of the transition>
' Block of properties of the transition node
FROM=...
TO=...
CONDITION=...
}
```

Properties of a Transition node

The following table lists all the properties that can be defined in a "Transition" node:

Table 14.8. Logical properties of a "Transition" node

Name of the property=Value	Description of the property	Example	Comment
FROM="<Name of the original page>"	Defines the name of the original page of the transition. "String" type property.	FROM="Page2"	This property is mandatory if the transition is defined inside the "Root" node and inapplicable if it is defined in a "Page" node, "Finish" or "Start" node.
TO="<Name of the target page>"	Defines the name of the target page of the transition. "String" type property.	TO="Page3"	This property is mandatory if the transition is defined inside a "Root" or "Page" node and inapplicable if it is defined in a "Finish" node.
CONDITION=<Script>	Defines the condition to be fulfilled to cause the transition. "Script" type property that returns a Boolean.	CONDITION= {Comment}="user"	This property is not available in "Start" nodes.
DO=<Script>	Defines a script to be executed at the time of the transition. "Boolean" type property.	{DO= Filter=""}	

Specificities of a Transition node

A "Transition" node does not have a sub-node.

Why define transitions in the "Root" node?

Taking transitions out of "Page" nodes allows you to create pages that can be used in any scripts and rationalizes the writing of scripts.

Definition of a Finish node

A "Finish" node describes the final transition: that which leads to the final page of the wizard. It is a specific type of "Transition" node that does not have "FROM" and "TO" properties. Apart from this exception, the syntax and properties in a "Finish" node are identical to those of a "Transition" node.

The CHAIN property, specific to the Finish node allows you to trigger the execution of another wizard.

Table 14.9. Logical properties of a "Finish" node

Name of the property=Value	Description of the property	Example	Comment
CHAIN=<SQL name of the wizard to be executed>	Defines the SQL name of the wizard to be executed at the end of the current wizard. If this property is not populated, no wizards will be executed. "String" type property	CHAIN = "Move"	
CONDITION=<Script>	Defines the condition validating the **Finish** button. "Boolean" type property.		
DO=<Script>	Defines the script to execute at the end of the wizard. "Script" type property that returns a Boolean.		

Note:

The PARAMS node allows you to pass parameters to the following wizard.

Table 14.10. Physical property of a "Finish" node

Name of the property=Value	Description of the property	Example
SUMMARY=<TRUE|FALSE>	Displays (=TRUE) or not (=FALSE) a summary page when launching the wizard. The amLog and amProgress functions enable you to populate this page.	
SHOWPROGRESS-BAR=<TRUE|FALSE>	Displays (=TRUE) or not (=FALSE) a progress bar in the summary page.	
SHOWLOG-LIST=<TRUE|FALSE>	Displays (=TRUE) or not (=FALSE) a progress log in the summary page.	
LABEL="Title"	Displays a title for the summary page.	
ISHTML=<TRUE|FALSE>	Defines the nature of the text of the caption. "Boolean" type property.	
TITLE="Title"	Defines a title for the summary page. The default title of the summary page is that of the root node.	

Definition of a Start node

A "Start" node describes how the wizard is started. It is a specific type of "Transition" node that does not have a "FROM" or "CONDITION" property. Apart from this exception, the syntax and properties in a "Start" node are identical to those of a "Transition" node.

Table 14.11. Logical property of a "Start" node

Name of the property=Value	Description of the property	Example

DO=<Script>	Defines the script to execute at start-up. "Script" type property that returns a Boolean.
TO="<Name of the starting page>"	Defines the name of the first page to display. "String" type property.

Note:

If this node does not exist, the wizard starts on the first page.

Definition of a Timer node

A "Timer" node enables you to perform a task at a regular interval.

Table 14.12. Logical property of a "Timer" node

Name of the property=Value	Description of the property	Example
AUTO= <TRUE|FALSE>	Defines whether the timer is automatically launched when the page is displayed. This property can be used to stop or restart a timer. "Boolean" type property.	
ENABLED= <TRUE|FALSE>	Defines whether the timer is enabled (=TRUE) or not (=FALSE). "Boolean" type property.	
INTERVAL=time interval	Defines the time interval between two executions of the timer. Duration expressed in milli-seconds.	

TIMER="Script"	Defines what must be executed each time the timer interval has passed. Script type property:
VALUE=tickcount	Number of times the interval has been passed. Each property dependent on this interval will be automatically reevaluated at each occurrence. This value (tickcount) is automatically incremented.

Definition of Long and String nodes

Long and String nodes define variables. These can be referenced in all nodes of the wizard. The name of the node determines the name of the variable.

These nodes only have one single property whose type depends on the node; its type is LONG for a Long node and STRING for a String node. This property, VALUE, allows you to define the value of the variable.

Table 14.13. Logical property of a Long or String node

Name of the property=Value	Description of the property	Example	Comment
VALUE=<Value>	Defines the value of the variable whose name is that of the node. "Long" type property for a Long node or "String" for a String node.	VALUE=12	

Note:

Long and String nodes can be defined in any node in the wizard. They do not have sub-nodes.

Definition of a Control node

Controls of a page interact with the user. You may define as many controls as you want for a given page. AssetCenter fully manages the organization of controls within a page. You do not have to specify the positioning of each of the controls that you define.

"Control" type nodes are exclusively made up of a block of properties applicable to a defined control.

General syntax of a Control node

The general syntax of a "Control" type node is as follows:

```
' Declaration of the control
{ <Control type> <Control name>
' Properties of the control
...
}
```

Types of controls and associated properties

All the controls have properties in common. However, there are properties that are specific to certain controls.

Common properties

The following table lists optional properties applicable to all controls:

Table 14.14. Logical properties common to all controls

Name of the property=Value	Description of the property	Example	Comment
MANDATORY=<TRUE|FALSE>	Forces the user to populate the control in order to validate a transition. This property is ignored if the control is: • not visible • read only • disabled	MANDATORY=TRUE	This property is not available for "CHECKBOX" and "LABEL" controls
VALUE=<Value>	Defines the default value of the control when created. <Value> depends on the control concerned. The type of this property depends on the type of control (Boolean, text, etc.).	For example, if it is a "CHECKBOX" control, <Value> can be "TRUE" or "FALSE".	
PERMANENT=<TRUE|FALSE>	When you go from one wizard page to the next, the controls are destroyed. Defines if the control is kept and hidden when going from one page to another (=TRUE) instead of being destroyed (=FALSE). "Boolean" type property.		

Name of the property=Value	Description of the property	Example	Comment
SERIALIZE=<TRUE|FALSE>	This property enables you to serialize the wizard at the level of the control. If the serialization of the root node is active (=TRUE), you can deactivate it at the level of the control.		By default, this property has the value of the SERIALIZE property at the root node.
HELP = "help"	This property enables you to include help text in HTML format into the control of a wizard. You access this help by pressing SHIFT+F1.		

Table 14.15. Physical properties common to all controls

Name of the property=Value	Description of the property	Example	Comment
VISIBLE=<TRUE|FALSE>	Defines whether the control is visible (=TRUE) or not (=FALSE). "Boolean" type property.	Label1.Visible=TRUE	
ENABLED = <TRUE|FALSE>	Defines whether the control is active (=TRUE) or not (=FALSE). "Boolean" type property. "Boolean" type property.	Choice1.Enabled=FALSE	

Name of the property=Value	Description of the property	Example	Comment
READONLY = <TRUE|FALSE>	Defines whether the value of the control is in read-only (=TRUE) and thus cannot be modified b the user, or whether it is editable (=FALSE).	READONLY=TRUE	
LABEL = "<Text of the label>"	Defines an optional text, displayed above or to the left of the control. "String" type property.	Choice1.Label="Select person"	This label supports HTML
LABELLEFT	This property enables you to put the control's label on its left. Using this property means that you must populate the XOFFSET property. "Boolean" type property.		
XOFFSET	Defines the space reserved for the control's label if it has been placed to the left of the control (by using the LABELLEFT property). **Twip** type property.		
ISHTML	Defines the nature of the text of the label. By default, the nature of the test is HTML. "Boolean" type property.		This label supports HTML

Name of the property=Value	Description of the property	Example	Comment
INDENT	This property enables you to shift the control and its caption to the right. **Twip** type property.		
YOFFSET	Defines an 'offset' before the control and its caption. **Twip** type property.		
YOFFSET2	Defines a 'offset' after the control and its caption. **Twip** type property.		

The CheckBox control

The "CHECKBOX" control defines a check box.

Properties

In addition to the optional properties common to all controls, the "CHECKBOX" control recognizes the following property:

Table 14.16. Property of "CHECKBOX" control

Name of the property=Value	Description of the property	Example
CAPTION="<Text>"	Defines the text of the check box. "String This text cannot be in HTML and must contain one single line. "String" type property.	TEXT="Identify by name"

The ComboBox control

The "COMBOBOX" control defines a single choice in an enumeration (itemized list) of predefined values.

Properties

In addition to the optional properties common to all controls, the "COMBOBOX" control recognizes the following property:

Table 14.17. Physical properties of the "COMBOBOX" control

Name of the property=Value	Description of the property	Example	Comment
VALUES="<Name=Value, Name=Value, Name=Value,...>"	Defines the value couples ("Name"= "Value") for the "Combo" control. "Name" defines the text that is displayed in the control, "Value" the value attributed if this "Name" is selected by the user. "String" type property.	VALUES="Table of assets=asset, User=user"	If you omit the "Value", AssetCenter will automatically assign one. **Example** VALUES = "A,B,C" is equivalent to VALUES = "A=1,B=2,C=3"

The OptionButtons control

The "OPTIONBUTTONS" control defines a group of option buttons (radio buttons).

Properties

In addition to the optional properties common to all controls, the "OPTIONBUTTONS" control recognizes the following properties:

Table 14.18. Physical properties of the "OPTIONBUTTONS" control

Name of the property=Value	Description of the property	Example
VALUES="<Title=Value, Title=Value, Title=Value,....>"	Defines the value couples ("Title"="Value") for the "CHOICE" control. "Name" defines the text of the option button, "Value" the value attributed to the control if this option button is selected by the user. "String" type property.	VALUES="Table of assets=asset, User=user"
BORDER=<TRUE|FALSE>	Specifies whether a border is drawn for the group of option buttons (=TRUE) or not (=FALSE) If the group of buttons is framed, the text is integrated into the border of the frame. This text cannot be in HTML or contain mutliple lines "Boolean" type property.	BORDER= TRUE

The ListBox control

This "LISTBOX" control defines a list of objects that can be selected. "LISTBOX" controls can be multi-column controls.

Properties

In addition to the optional properties common to all controls, the "LISTBOX" control recognizes the following properties:

Table 14.19. Physical properties of the "LISTBOX" control

Name of the property=Value	Description of the property	Example	Comment
LISTHEIGHT = <Percentage>	Defines size of the "LISTBOX" control in relation to other "LISTBOX" controls in the wizard as a whole. "Long" type property.	LISTHEIGHT=50	If there are two "LISTBOX" controls with values "10" and "20" respectively for this property, the second control will be twice as high as the first one.
MULTISEL = <TRUE|FALSE>	Specifies whether the control supports multiple-selection (=TRUE) or not (=FALSE). "Boolean" type property.	MULTISEL=TRUE	
COLTITLE = "<Column|Column|Column...>"	Defines the title of the columns in the list. Replace "Column" by the title of the column. "String" type property.	COLTITLE = "Name|FirstName"	
COLWIDTH = "<Width|Width|Width...>"	Defines the size of the columns proportionally to the overall size of the control. "String" type property.	COLWIDTH = "50|50"	

Name of the property=Value	Description of the property	Example	Comment
VALUES = "<Texte|Texte|...= Value, Text|Texe|...= Value,...>"	Defines value couples ("Text|Text|..."= "Value") for the "LISTBOX" control. "Text|Text|.." defines the text to be displayed in each of the columns for a line of the "LISTBOX" control, "Value" the value attributed to the control if this line is selected by the user. "String" type property.	VALUES="Table of assets=asset, , User=user,"	If you omit the "Value", AssetCenter will automatically assign one. VALUES="A,B,C" is equivalent to VALUES="A=1,B=2,C=3" **Example** This property can be populated directly or using the AmdbGetList, function by writing, for example: VALUES = AmDbGetList ("SELECT Name, FirstName FROM amEmplDept WHERE Name Like 'A%'", "|", ",", "=") Do not confuse the "VALUES" and "VALUE" properties.
EDITABLE="<0|1>"	Defines whether the text in the columns can be edited or not. "String" type property.	EDITABLE="0|1"	
TABLE="<Name of the table>"	Defines the application context of the column titles if they have been defined for the "COLNAME" property.	TABLE="amEmplDept"	

Name of the property=Value	Description of the property	Example	Comment
COLNAME="<name|name|name>"	Defines the title of the properties of the columns in the list with the SQL field names. The "TABLE" property must be populated; Replace "name" by the SQL name of the field for the title of the column. If the title of the column has been populated with the "COLTITLE" property, this takes priority over the "COLNAME" property, but it keeps the type of the field (text, date, etc.).	COLNAME="Name|FirstName|dtHire"	
MULTISEL = <TRUE|FALSE>	Defines the use of multiselection for a list. "Boolean" type property.	MULTISEL=1	

Table 14.20. Methods of the "LISTBOX" control

Name of the property=Value	Description of the property	Example	Comment
VALUES (i)	Returns the contents of the line (i).	a|b|c	
VALUES.COUNT()	Calculates the number of lines in the "VALUES" property.	retval = {listbox1.values.count()}	
VALUES.CELL(h,v)	Returns the contents of the cell designated by its coordinates (horizontal, vertical).	VALUES.CELL(2,4)	

Name of the property=Value	Description of the property	Example	Comment
VALUES.COLUMN(i)	Returns the contents (value) of the column (i). If i=0 or is empty, this instruction returns the IDs of the column.	VALUES.COLUMN(1)	
VALUES.SORT(iCol, bAsc)	Sorts the column (i) in an ascending order or not (bAsc=1 or bAsc=0).	{ LISTBOX lb VALUES = "first,second,third" } { COMMANDBUTTON btn { CLICK = RetVal = {lb.Values.Sort(1)} } }	
VALUES (i,0) ID(i)	Returns the value of the ID of the line (i).		

Table 14.21. Mandatory logical property of the "LISTBOX" control

Name of the property=Value	Description of the property	Example
TABLE=<Name of the table>	Name of table used to extract the titles of columns. "String" type property.	TABLE=**amAsset**
COLNAME=<Title|Title|...>	Defines the titles of columns, using the SQL name of fields in the table defined using the "TABLE" property. This property also allows you to define the edit controls used. The control is the same as is used in AssetCenter to populate the field. AssetCenter will first take the values of the "COLTITLE" property (if this exists) into account to define the titles of columns. "String" type property.	COLNAME="Name|FirstName"

The Label control

The "LABEL" control simply defines a label. This control has the following property:

Table 14.22. Physical properties of the "LABEL" control

Name of the property=Value	Description of the property	Example	Comment
CAPTION=<Text>	Contains the text displayed in the label.	CAPTION="Select a location"	

The ProgressBar control

The "PROGRESSBAR" control defines a progress bar.

Properties

In addition to the optional properties common to all controls, the "PROGRESSBAR" control recognizes the following properties:

Table 14.23. Physical properties of the "PROGRESSBAR" control

Name of the property=Value	Description of the property	Example
MAXVALUE=<Maximum value>	Defines the maximum value corresponding to 100% of the progress bar. The "VALUE" property indicates the current value of the control. "Long" type property.	MAXVALUE=200

The CommandButton control

The "COMMANDBUTTON" control defines an action button.

Properties

In addition to the optional properties common to all controls, the "COMMANDBUTTON" control recognizes the following properties:

Table 14.24. Physical properties of the "COMMANDBUTTON" control

Name of the property=Value	Description of the property	Example
WIDTH=<Width>	Defines in twips the width of the button. "Long" type property.	WIDTH=250
HEIGHT=<Height>	Defines in twips the height of the button. "Long" type property.	HEIGHT=125
CAPTION=<Text>	Defines the text (non HTML) displayed within the button. "String" type property.	CAPTION="Start"
CLICK=<Basic script>	Defines the Basic script that is executed when the user clicks the button.	

The DBListBox control

The "DBLISTBOX" control defines a list of records that can be selected from the database. This control can be a multiple-column control. The list displayed in the control is the result of a partial AQL query (only the WHERE clause is used) on the AssetCenter database.

Note:

The "VALUE" property returns the list of identifiers ("Id") of the selected lines. You cannot access the values of cells in the list. For this, you need either to perform another query, or use a "LISTBOX" type control.

Properties

In addition to the optional properties common to all controls, the "DBLISTBOX" control recognizes the following properties:

Table 14.25. Physical properties of the "DBLISTBOX" control

Name of the property=Value	Description of the property	Example	Comment
TABLE="<SQL name of the table>"	Defines the table concerned by the query. "String" type property.	TABLE = amAsset	This property is mandatory.
COLNAME="<SQL name of the field or link|SQL name of the field or link|...>"	Defines the data items to be extracted from the database (identified using their SQL name). "String" type property.	COLNAME = "Name|FirstName"	
COLWIDTH = "<Width|Width|Width|...>"	Defines the width of the columns, as a percentage of the overall size of the "DBLISTBOX" control. "String" type property.	COLWIDTH="40|60"	
LISTHEIGHT = <Percentage>	Defines the size of the "DBLISTBOX" control in relation to other "DBLISTBOX" controls in the wizard as a whole. "Long" type property.	LISTHEIGHT=50	If there are two "DBLISTBOX" controls with values "10" and "20" respectively for this property, the second control will be twice as high as the first one.
TREE=<TRUE|FALSE>	Displays the data in tree mode (=TRUE) or not (=FALSE). "Boolean" type property.	TREE=TRUE	By default, this property is set to "FALSE"

Name of the property=Value	Description of the property	Example	Comment
MULTISEL = <TRUE|FALSE>	Specifies whether the control supports multiple-selection (=TRUE) or not (=FALSE). "Boolean" type property.	MULTISEL=TRUE	
DBLCLICK = <TRUE|FALSE>	If this property is set to TRUE, AssetCenter will simulate clicking the **Next** button of the current page each time you double-click a row.	DBLCLICK=FALSE	
FILTER= "<Condition>"	Defines the AQL "WHERE" condition to filter records to be processed in the query. "String" type property.	FILTER= "User.lEmplDeptId='Colombo, Gerard' "	
MAXSEL = <TRUE|FALSE>	Defines the possibility of selecting more than 99 objects (=TRUE) or not (=FALSE). The selection is limited to 99 objects by default (=TRUE).		
VALUES.SORT(iCol, bAsc)	Sorts the column (iCol) in an ascending order or not (bAsc=1 or bAsc=0).	VALUES.SORT(2, 0)	

Name of the property=Value	Description of the property	Example	Comment
[Value.]ISSELECTION()	If the selection contains values other than 0 (in other words, the user did not make a selection in the list), this method returns a non-null value.	MANDATORY = not {dblistbox1.IsSelection()} (A transition is impossible if the user has not selected anything in the list.)	
TABLE.LABEL([iNameType])	This method returns the name of the table on which the control is defined. The name types are the following: • 1 - System name • 2 - SQL name • 3 - Default label • 4 - Description (text in help) The "TABLE" property needs to have been populated.	RetVal = {dblistbox1.table.label(2)}	

The DBQueryBox control

The "DBQUERYBOX" control defines a list of records that can be selected. This control can be a multi-column control. The list displayed in the control is the result of a full AQL query on the AssetCenter database.

Properties

In addition to the optional properties common to all controls, the "DBQUERYBOX" control recognizes the following properties:

Table 14.26. Physical properties of the "DBQUERYBOX" control

Name of the property=Value	Description of the property	Example	Comment
QUERY="<Full AQL query>"	Defines the AQL query that returns the information to be displayed in the "DBLIST" control. "String" type property.	QUERY="SELECT Name, FirstName FROM amEmplDept WHERE Location=Ariane Building'"	
COLTITLE="<Column|Column|...>"	Defines the title of columns in the list. "String" type property.	COLTITLE = "Name|FirstName"	
COLWIDTH="<Width|Width|...>"	Defines the size of columns in the list as a percentage of the overall size of the control. "String" type property.	COLWIDTH = "50|50"	
LISTHEIGHT=<Percentage>	Defines the size of the "DBQUERYBOX" control proportionally to other "DBQUERYBOX" control in the wizard as a whole. "Long" type property.	LISTHEIGHT=50	If there are two "DBQUERYBOX" controls with values "10" and "20" respectively for this property, the second control will be twice as high as the first one.

Name of the property=Value	Description of the property	Example	Comment
TREE=<TRUE|FALSE>	Displays the data in tree mode (=TRUE) or not (=FALSE). "Boolean" type property.	TREE=TRUE	By default, this property is set to "FALSE"
MAXSEL = <TRUE|FALSE>	Defines the possibility of selecting more than 99 objects (=TRUE) or not (=FALSE). The selection is limited to 99 objects by default (=TRUE).		
MULTISEL=<TRUE|FALSE>	Specifies whether the control supports multiple-selection (=TRUE) or not (=FALSE). "Boolean" type property.	MULTISEL=TRUE	
DBLCLICK=<TRUE|FALSE>	If this property is set to TRUE, AssetCenter will simulate clicking the **Next** button of the current page. "Boolean" type property.	DBLCLICK=FALSE	

Name of the property=Value	Description of the property	Example	Comment
[Value.]ISSELECTION()	If the selection contains values other than 0 (in other words, the user did not make a selection in the list), this method returns a non-null value.		

The DBEdit control

The "DBEDIT" control creates a control identical to that used to populate a field in the AssetCenter database. The control differs according to each field type (date, monetary, etc.).

Note:

The magnifier button in this control allows you to select values that are effectively in the database, but you can enter another value.

For this control, the "VALUE" property has the "Variant" (it depends on the control).

Properties

In addition to the optional properties common to all controls, the "DBEDIT" control recognizes the following properties:

Table 14.27. Physical properties of the "DBEDIT" control in "Normal" mode

Name of the property=Value	Description of the property	Example
TABLE="<SQL name of the table>"	SQL name of the starting table. "String" type property.	TABLE=amAsset

Name of the property=Value	Description of the property	Example
FIELD=<SQL name of the field>	AQL name of the field used for the control. "String" type property.	FIELD="seAcquMethod"
TABLE.LABEL([iNameType])	This method returns the name of a data table. The name types are the following: • 1 - System name • 2 - SQL name • 3 - Default label • 4 - Description (text in help) The properties "TABLE" and "NAME" need to have already been populated.	→ Physical properties of the "DBQUERYBOX" control [page 280]
FIELD.LABEL([iNameType])	This method returns the name of a given field. The name types are the following: • 1 - System name • 2 - SQL name • 3 - Default label • 4 - Description (text in help) The "TABLE" property needs to have been populated.	

The DBTable control

The "DBTABLE" control creates a control for entering a table in the AssetCenter database.

Properties

This control does not have any additional properties.

The DBPath control

The "DBPATH" control creates a control for entering a path of the AssetCenter database.

Properties

In addition to the optional properties common to all controls, the "DBPATH" control must have the following mandatory property:

Table 14.28. Mandatory logical property of the "DBPATH" control

Name of the property=Value	Description of the property	Example
TABLE="<SQL name of the table>"	Name of the table from which you want to select a path. "String" type property.	TABLE=**amAsset**

The LinkEdit control

The "LINKEDIT" control creates a control for entering a link in the AssetCenter database.

Properties

In addition to the optional properties common to all controls, the "LINKEDIT" control has the following properties:

Table 14.29. Logical property of the "LINKEDIT" control

Name of the property=Value	Description of the property	Example / Comment
TABLE=<SQL name of the table>	Name of the table in which you want to select a link. "String" type property.	TABLE="amAsset"
FILTER=<WHERE clause of an AQL query>	Defines an AQL filter. "String" type property.	This property is optional.

Name of the property=Value	Description of the property	Example / Comment
LINK=<SQL name of the link>"	SQL name of a link from the table defined in the TABLE property. Optional property.	LINK="POrLIne"
ZOOM=<TRUE|FALSE>	Displays (=TRUE) or not (=FALSE) the magnifying tool. This property only applies if the wizard is not modal (MODAL=FALSE property at the root node).	
SRCCHOICE=<TRUE|FALSE>	Displays (=TRUE) or not (=FALSE) the  icon. This property only applies if the wizard is not modal (MODAL=FALSE property at the root node).	
TABLE.LABEL([iNameType])	This method returns the name of the source table of the link. The name types are the following: • 1 - System name • 2 - SQL name • 3 - Default label • 4 - Description (text in help) The "TABLE" property needs to have been populated.	→ Physical properties of the "DBQUERYBOX" control [page 280]

Name of the property=Value	Description of the property	Example / Comment
LINK.LABEL([iNameType])	This method returns the name of the link. The name types are the following: • 1 - System name • 2 - SQL name • 3 - Default label • 4 - Description (text in help) The "TABLE" and "LINK" properties need to have already been populated.	

The TextBox control

The "TEXTBOX" control creates a control for entering text.

Properties

In addition to the optional properties common to all controls, the "TEXTBOX" control can have the following property:

Table 14.30. Physical property of the "TEXTBOX" control

Name of the property=Value	Description of the property	Example
MULTILINE=<number>	This property is set to "0" if the "TEXTBOX" control is mono-line, or a numeric value expressing the percentage of the displayed height of the control if the control is multi-line.	MULTILINE=50
PASSWORD=<TRUE|FALSE>	These properties hide (=TRUE) or display (=FALSE) the typed text.	

The CONTROL control

The "CHART" control enables you to display a graphic. This can be composed of several series.

Properties

In addition to the optional properties common to all controls, the "CHART" control can have the following properties:

Table 14.31. Logical properties of the "CHART" control

Name of the property=Value	Description of the property	Example
SERIES	Defines the list of line names, separated by a vertical bar. This list must not be empty, otherwise nothing will appear in the graphic.	SERIES="purchase price|sale price"
VALUES	Defines the numeric values of the lines in the graphic. Two-dimensional table	VALUES="1|2,1|4"
FORMAT	Defines the data type: • Long integer (long) • Double • Number (number) • Percentage (percent)	
SERIE	Number of the line you clicked. The CHART property must be in interactive mode (=TRUE).	
INDEX	Number of the column you clicked on. The CHART property must be in interactive mode (=TRUE).	
CLICK	Calls on this property's script when you click the graphic.	

Table 14.32. Physical properties of the "CHART" control

Name of the property=Value	Description of the property	Example
MODE	Defines the graphic type: • MODE=0: vertical bars • MODE=1: horizontal bars • MODE=3: segments	
LABELS	Defines the title of columns.	january|february
3D	Defines whether the graphic is 3D (=TRUE) or not (=FALSE).	
COLORS	Defines the list of colors for each line. The objects in the list are separated by a vertical bar. RGB value in decimal.	255|16777215|16711680 Display the colors blue, black and red.
INTERACTIVE	Defines if the graphic is interactive (=TRUE) or not (=FALSE) (in other words, if it's active when the cursor passes over it).	
POPUP	Displays (=TRUE) or not (=FALSE) the shortcut menu.	
BACKGROUND	Defines whether there is a background (=TRUE) or not (=FALSE) for the image.	
BACKIMAGE	Defines the path of the image used as a background for the graphic. The "BACKGROUND" property must by validated (=TRUE) in order to display a background image.	
STACKED	Defines whether the bars of the graphic are stacked (=TRUE) or not (=FALSE).	
CHARTHEIGHT	Defines the size of the "CHART" control in relation to other controls in the wizard as a whole.	
CAPTION	Displays the title.	
ELEVATION	Defines the elevation in degrees of the 3D for pie chart.	
ROTATION	Defines the angle of rotation for the pie chart. Value expressed in degrees.	

Name of the property=Value	Description of the property	Example
DISPLAYLABELS	Displays (=TRUE) or not (=FALSE) the titles of the columns (LABELS).	
DISPLAYSLE-GEND	Displays (=TRUE) or not (=FALSE) the key of the lines.	

The FILEEDIT control

This control displays a dialog box enabling you to save or load a file or folder.

Table 14.33. Properties of the "FILEEDIT" control

Name of the property=Value	Description of the property	Example
OPENMODE	Defines the type of dialog box. • OPENMODE=1: opens a file. • OPENMODE=2: saves a file. • OPENMODE=4: opens a folder. • OPENMODE=8: saves a folder.	
FILTERS	Defines the criteria for displaying the files listed in the dialog box.	(*.txt)|*.txt|(*.scn)|*.scn|
DEFEXT	Defines the default file name extension.	(*.scn)|*.scn|

The TICKEDIT control

This control enables you to insert a timer.

Table 14.34. Properties of the TICKEDIT control

Name of the property=Value	Description of the property	Example
VALUE	Representation (in a string form) of the parameters defined by the user in the timer.	

Name of the property=Value	Description of the property	Example
LISTHEIGHT	Defines the size of the "TICKEDIT" control in relation to other controls in the wizard as a whole. "Long" type property.	

The CALENDAR control

This control enables you to insert a calendar.

The TIMSPANEDIT control

This control enables you to insert a time-span edit zone.

The NUMBOX control

This control enables you to insert a numeric type control.

Table 14.35. Properties of the NUMBOX control

Name of the property=Value	Description of the property	Example
MINVALUE	Defines the minimum value of the number. Infinite value by default.	
MAXVALUE	Defines the maximum value of the number. Infinite value by default.	
FORMAT	Defines the format of the number: • LONG in the format defined in the Control Panel of your operating system. • RAWLONG • DOUBLE in the format defined in the Control Panel of your operating system.	

The COMBOEDIT control

This control enables you to insert a drop-down list type control.

Table 14.36. Properties of the COMBOEDIT control

Name of the property=Value	Description of the property	Example
VALUES	Defines the value couples "Name" defines the text that is displayed in the control, "Value" the value attributed if this "Name" is selected by the user. "String" type property.	

The DATETIMEEDIT control

The DATETIMEEDIT control

Table 14.37. Properties of the control

Name of the property=Value	Description of the property	Example
FORMAT	Defines the format of the control: • Date • Time • Date and time The format of the control depends on the system parameters defines by the user.	2002/02/07 13:37:19 2002/02/07 13:37:19

Using the graphical editor

AssetCenter makes it possible for you to create wizards using an integrated graphical editor. This editor aims to simplify and speed up the process of creating a wizard. It is not meant as a substitute for the scripting language, the understanding of which is essential in order to design wizards.

- Overview of the interface
- Creating a new node
- Editing the properties of a node
- Compiling, executing and debugging a wizard

Note:

To use the graphical editor, the action being created or modified must by a "Wizard" type action (SQL name: seActionType).

Overview of the interface

To access the graphical editor, select the **Tools/ Actions/ Edit** menu item. The graphical editor is shown in the **Wizard** tab of the action detail. It is made up of three parts:

- A toolbar containing the most common functions.
- A **Hierarchy** section that shows a tree-view of the structure of the wizard.
- A section that lists the properties of the node selected in the hierarchy.

Toolbar

The toolbar allows you to directly activate edit-commands. If you leave the mouse pointer over a ToolTip for a short period of time, a ToolTip is displayed describing the icon.

Edit commands

Four edit commands are available:

- switches the editor between text and graphical mode.
- moves the node up one level inside its own parent node.
- moves the node down one level inside its own parent node.
- deletes the selected node.

Execution and debugging commands

These commands allow you to compile, debug and execute the script:

Figure 14.2. Execute and debug button

Search tool

The toolbar includes a search tool that can be used to search a character string in the tree-structure of the wizard ("Ctrl+F" puts you directly in this control):

Click this zone and enter the text to search. If successful, AssetCenter moves the selection to the occurrence found ("F3" and "Shift+F3" keyboard shortcuts search the next and previous occurrences respectively).

Note:

In text mode, the search is full text. In graphic mode, the search only concerns the name of a property.

Tree-view of the wizard

The left part of the graphical editor shows a tree-view of the wizard:

When you select a node in the tree, AssetCenter lists the properties associated with this node in the right part of the screen.

List of properties corresponding to the selected node.

The right part of the screen allows you to enter values for the properties of a node:

Each property has a fixed value or a script. The following color codes are used:

- When a property uses its default value, its name and its value are displayed in gray. You can force another value for this property. It will be displayed in black.
- When a property uses a user-defined value or a script, its name and value are displayed in black.
- When a property is mandatory, its name and value are displayed in red.
- The modified values are displayed in blue.

Creating a new node

This section details the operations you can perform on a node. The toolbar allows you to move the node up or down or delete it. Here we will deal with the creation of a new node.

Note:

You can also move a node up or down or delete it using the shortcut menu. In this case, right-click the selected node.

To create node, first select its parent node. For example, to create a new "Page" node, you must first click the "Root" node. When you have selected the parent node, right-click to open the shortcut menu. The "New" menu item groups together the nodes you can create:

AssetCenter inserts the node in the tree of the wizard.

Editing the properties of a node

Once the node is created, you can attribute values to the properties of this node. You can do this in the right part of the editor.

The value of a property can be defined in two ways:

- By entering a fixed value
- By defining a script

Note:

A script always takes precedence over a fixed value. If you assign both a script and a value to a property, AssetCenter will ignore the fixed value and interpret the script.

Assigning a fixed value to a property

Click the "Value" column of the concerned property directly. According to the type of data accepted by the property (Text, Boolean, Double precision number, etc.), AssetCenter lets you select from a list of possible values or populate a text edit zone.

Assigning a script to a property

Select the property to which you want to assign a script. The script itself is entered in the **Script** field under the list of properties.

Note:

By selecting **Restore default value** in the shortcut menu (right-click a property), AssetCenter cancels the fixed value or script and resets the property with its default value. This operation is only possible for properties for which a value or script has be defined by the user (these properties are displayed in black).

Compiling, executing and debugging a wizard

You can launch the wizard by clicking the button in the toolbar of the editor. Any errors encountered while executing are displayed in the error history window (accessible via the integrated wizard debugger. By pressing Shift+F9 you can interrupt execution (if the wizard is modal) and activate the debugger.

In this way, you can easily repair and correct errors in the wizard.

Note:

The execution button is not available if the wizard is contextual.

Example of creating a wizard

To illustrate the theoretical part of programming a wizard, we will create a "Move" wizard. In association with a "Database" type action, it simplifies the process o moving a user and associated assets from one location to another. The creation of this wizard is described step by step. We invite you to create this wizard by yourself an to consider this section as a guide in case of problem.

1 Example of creating a wizard
2 Step #1- Analyze your needs
3 Step #2- Define how the wizard is organized

Example of creating a wizard

The objective of this wizard is move assets from one location to another. For this we need:

1 To identify the assets to be moved.

2 Choose the new location for these assets.

How to identify the assets to be moved

There are three ways to identify the assets to be moved:

- Identifying them according to their user. Once this user is selected, you will need to select the assets to be moved.
- Identifying the assets to be moved directly by selecting their records in the table of assets.
- Identifying the assets to be moved according to their location. You first select a location, then the asset from this location to be moved.

Note:

We will therefore have to create a user-choice page in order for the user to select a method of selection for the assets to be moved.

Choosing a new location

To choose a new location for the assets, simply select a record in the table of locations.

Step #1- Analyze your needs

According to the needs defined in step 1, we need to define the organization of the wizard. I.e.:

1 The number of pages.

2 How the different pages are linked.

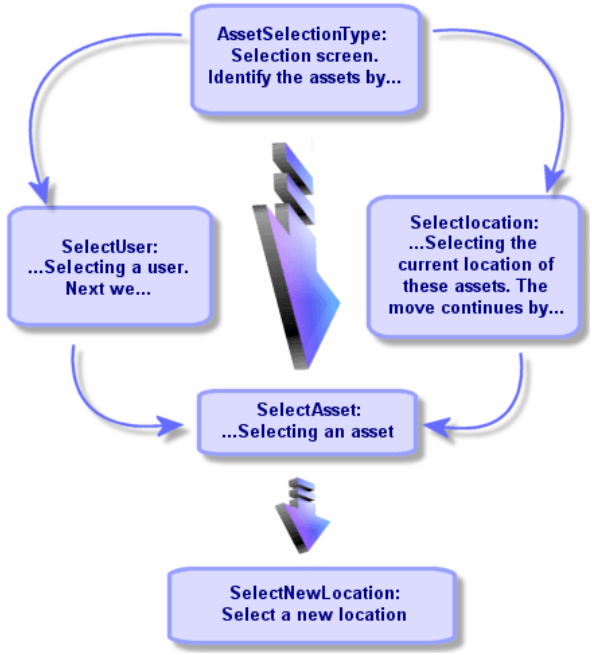
3 The contents of each of these pages.

Note:

We know since step 1 that we need to create a selection page. This page will be the first page of the wizard. We will call it "AssetSelectionType".

Now we define how the wizard is organized using the following diagram:

Figure 14.3. Wizard - organization example



Using this flow-chart, we can now define the transitions page by page:

Page	Can lead to pages
AssetSelectionType	SelectAsset, SelectUser, SelectLocation
SelectAsset	SelectNewLocation
SelectUser	SelectAsset
SelectLocation	SelectAsset
SelectNewLocation	None

Next we define the contents of the pages. This means the controls that allow the user to make choices:

Page	Objective of this page?	Control to use
AssetSelectionType	Allows the user to choose between three possibilities	A "CHOICE" control

Page	Objective of this page?	Control to use
SelectAsset	Allows the user to choose assets from a list of records of the table of assets.	An "ADBLIST" control
SelectUser	Allows the user to select a user from the table of departments and employees whose assets are to be moved.	An "ADBLIST" control
SelectLocation	Allows the user to select a current location in the table of locations.	An "ADBLIST" control
SelectNewLocation	Allows the user to select a new location in the table of locations.	An "ADBLIST" control

Step #2- Define how the wizard is organized

This step consists of writing the script of the wizard. In order to do this, use the descriptions of the structure of each of the nodes of the wizard. The commented source code of the Move wizard is listed below. This code represents but one way of writing the wizard. There are many other ways of writing a wizard performing the same task.

```
================================
;(c) Peregrine Systems 1999
;=================================================================
NAME = "Move"
TITLE = "Move user"
VERSION = "699"
;=============================================================================
===========
;Ask which user to move. By default, use selection in amEmplDept if conte
xt is on this table
;=============================================================================
===========
{ PAGE pgUser
  TITLE = "Choose the persons who are moving"
  { DBLISTBOX Users
    COLNAME = "Name|FirstName"
    COLWIDTH = "50|50"
    DBLCLICK = 1
    LABEL = "Persons to move"
    MULTISEL = 1
    TABLE = "amEmplDept"
    { VALUE =
      if [CurrentTable] = "amEmplDept" then
        RetVal = [CurrentSelection]
```

```
      else
        RetVal = ""
      end if
    }
    VISIBLE = 1
  }
  { TRANSITION trPersonToNewLoc
    TO = "pgNewLoc"
  }
}

;=====================================================================
==========
;Ask for new location
;=====================================================================
==========
{ PAGE pgNewLoc
  TITLE = "Choose the new location"
  { STRING UserName
    VALUE = AmDbGetString("SELECT FirstName + ' ' + Name FROM amEmplDept
WHERE lEmplDeptId IN (" & {pgUser.Users} & ")" )
  }
  { LABEL LABEL1
    CAPTION = "User(s) : " & {UserName}
  }
  { DBLISTBOX NewLocId
    COLNAME = "Name"
    COLWIDTH = "100"
    DBLCLICK = 1
    TABLE = "amLocation"
    VALUE = "-1"
  }
  { TRANSITION trNewLocToAssets
    TO = "pgRecap"
  }
}

;=====================================================================
==========
;Recapitulation
;=====================================================================
==========
{ PAGE pgRecap
  TITLE = "Recapitulation"
  { LISTBOX Users
    COLTITLE = "Name"
    COLWIDTH = "100"
    LABEL = "Persons to move"
    MANDATORY = 0
    MULTISEL = 1
    READONLY = 1
    VALUE = ""
    VALUES = AmDbGetList("SELECT FullName FROM amEmplDept WHERE FullName
LIKE LikeParam(amEmplDept_2:FullName)+'%' AND amEmplDept_2:lEmplDeptId IN
```

```
(" & {pgUser.Users} & ")","|",",", "=")
  }
}

;=====================================================================
==========
;Finish
;=====================================================================
==========
{ FINISH FINISH
  { DO =
    On Error Goto ErrHandler
    Dim lErr as long

    dim hRecord as Long

    dim iEmplCount as Integer
    iEmplCount  = {pgRecap.Users.VALUES.Count()}
    dim iMax as Long
    iMax = iEmplCount

    dim lLocaId as long
    lLocaId = {pgNewLoc.NewLocId}

    lErr = amStartTransaction()

    dim i as Integer
    For i = 1 To iEmplCount
      lErr = AmProgress((100 * i ) / iMax)
      lErr = AmLog("Moving the employee" + {pgRecap.Users.VALUES(i,1)})
      hRecord = AmGetRecordFromMainId("amEmplDept", {pgRecap.Users.VALUES
(i,0)} )
      If hRecord <> 0  then
        lErr = AmSetFieldLongValue( hRecord, "lLocaId", lLocaId)
        lErr = AmUpdateRecord(hRecord)
        lErr = AmReleaseHandle(hRecord)
      End If
    Next i

    lErr = amCommit()

    RetVal = 0
    Exit Function


    ErrHandler:
      On Error Goto 0
      AmLog(AmLastError() & " - " & AmLastErrorMsg())
      AmLog("The transaction has been cancelled")
      RetVal = 1
      Exit function
  }
  SUMMARY = 1
}
```

Example of creating a query wizard (QBE)

A query wizard is a special kind of wizard that prefilters a given screen.

The following example explores and explains one of the standard query wizards.

Step 1 - Idenifying the needs

The aim of this wizard is to prefilter the work orders screen.

We must therefore:

1 Determine which fields are going to be used in the query wizard.

2 Determine the type of the field.

Step n°2 - Designing the wizard

According to the needs defined in step 1, we need to define the organization of the wizard. I.e.:

1 The number of pages.

In order not to overload the filtering step, we recommend only creating one page.

2 The contents of each of these pages.

3 How the different pages are linked.

Page	Objective of this page?	Which controls to be used?
Filter	Presents the different fields enabing the screen to be pre-filtered	TEXTBOX COMBOBOX LABEL LINKEDIT DATETIMEEDIT CHECKBOX
Advanced	Presents the optional fields, which enable a more detailed prefiltering	COMBOBOX LINKEDIT DATETIMEEDIT LABEL

Step 3 - Retranscribe the structure of the wizard using the script language

This section explains certain portions of the wizard script used for the work orders screen.

These portions can be used as the basis for new QBE wizards.

Important:

When defining the names of the objects in your wizard, do not use the reserved words in the wizard syntax (such as 'name' or 'title').

- ```
  NAME = "searchamWorkOrder"
  TITLE = "Filter the work orders"
  VERSION = "3152"
  { PAGE Filter
    TITLE = "Quick search"
  ```

  Definition of the title page of the wizard.

- ```
  { TEXTBOX WONo
      LABEL = AmGetFieldLabelFromName("amWorkOrder","WONo")
      LABELLEFT = 1
      VALUE = ""
      XOFFSET = 1500
    }
  ```

  Create the **Work order number** object in the work orders table and display its title in the wizard.

- ```
  { COMBOBOX seStatus
      LABEL = AmGetFieldLabelFromName("amWorkOrder","seStatus")
      LABELLEFT = 1
      VALUE = ""
      { VALUES =
        Dim strFormat As String
        strFormat = AmGetFieldFormat(AmGetFieldFromName(AmGetTableFromNa
  me("amWorkOrder"), "seStatus"))
        RetVal = SysEnumToComboBox(strFormat)
      }
      XOFFSET = 1500
    }
    { COMBOBOX Priority
      LABEL = AmGetFieldLabelFromName("amWorkOrder","Priority")
      LABELLEFT = 1
      VALUE = ""
      { VALUES =
        Dim strValues As String
        strValues = AmDbGetList("SELECT Value FROM amItemListVal WHERE I
  temizedList.Identifier = 'amWOPriority'", "", ",", "")
  ```
  ```

```
      RetVal = EnumToComboBox(strValues)
    }
    XOFFSET = 1500
  }
```

Definition of itemized lists

1. For the **Status** object:
   1. Use the title of the **Status** field from the work orders table to display it in the wizard
   2. Define the field as a text type field
   3. Create the itemized list type object
   4. Use the contents of the system itemized list associated with this field and format using the **SysEnumToComboBox** function defined in the wizard-configuration library.
2. For the **Priority** field:
   1. Define the field as a text type field
   2. Use the contents of the itemized list associated with the **Priority** field
   3. Create the itemized-list type object and format the contents of this object using the **EnumToComboBox** function defined in the wizard-configuration library.

-
```
{ LINKEDIT Model
    FILTER = "Nature.seBasis = 3"
    LABEL = AmGetFieldLabelFromName("amWorkOrder","Model")
    LABELLEFT = 1
    TABLE = "amModel"
    VALUE = 0
    XOFFSET = 1500
    YOFFSET = 0
    YOFFSET2 = 0
  }
```

Create a control object for a link:

1. Create a filter on all models whose nature creates a work order
2. Use the title of the **Model** field in the work orders table to display it in the wizard
3. Define the table in which the link is selected

-
```
{ LABEL Empty2
  }
  { COMBOBOX OPdtResolLimit
    LABEL = AmGetFieldLabelFromName("amWorkOrder","dtResolLimit")
    LABELLEFT = 1
    VALUE = ""
```

```
    VALUES = "Before=<\=,After=>\=,Le=\=,After=<>"
    XOFFSET = 1500
  }
  { DATETIMEEDIT dtResolLimit
    FORMAT = "DateTime"
    LABEL = ""
    LABELLEFT = 1
    VALUE = ""
    XOFFSET = 1500
  }
```

Creation of an itemized list type object:

1 Use the title of the **Expected resol.** field from the work orders table to displayed it in the wizard

2 Define the list of possible operators

Creation of a date type object

```
{ CHECKBOX bAdvanced
    CAPTION = "Advanced search"
    VALUE = 0
  }
  { TRANSITION Transition
    CONDITION = 1
    { TO =
      If {Filter.bAdvanced} = 0 Then
        RetVal = "Finish"
      Else
        RetVal = "Advanced"
      End If
    }
  }
}
```

Creation of a check box:

1 Definition of the caption of the check box

2 Definition of the behavior of the check box: Cleared by default

3 Definition of the transition: If the check box is selected, the next button of the wizard is activated

```
{ FINISH FINISH
  { DO =
    Dim strFilter As String

    Dim strOperator As String
    strOperator = ""

    'WONo condition
    If {Filter.WONo} <> "" Then
      strFilter = strFilter & " AND WONo LIKE " & AmSqlTextConst({Filt
er.WONo})
    End If
```

```
    'Model condition
    If {Filter.Model} <> "0" Then
      strFilter = strFilter & " AND lModelId = " & {Filter.Model}
    End If

    If strFilter<> "" then
      strFilter = RightPartFromLeft(strFilter, "AND", 0)
      RetVal = AmOpenScreen("amWorkOrder","",strFilter,1,"")
    Else
      'ERR CANCEL
      RetVal = 2
    End if
  }
}
```

Create filter conditions (FINISH):

1 The FINISH object is created.

   This object is used to write the WHERE section of a SQL statement.

2 For each object a filter condition is defined:
   - **WONo condition** text field type object:
     1 If the field is not empty, filter according to the inserted value.
   - Link type object **Model condition** :
     1 If the field is not empty, filter according to the identifier of the link and display the links of the models.

3 A filter condition is defined for the value of the screen to be displayed.

   This condition uses the defined WHERE clause and only takes into account the portion starting after the first AND (in order to simplify the query).

4 The screen that is filtered is called by the **AmOpenScreen** function.

Frequently asked questions

This chapter aims to answer the questions you are likely to ask when creating a wizard.

Question

The following code does not work:

```
{lbxMyListBox.Values.Count}
```

Answer

You must enter opening and closing parentheses in the syntax of the method. Here is the corrected code:

```
{lbxMyListBox.Values.Count()}
```

Question

The following code does not work:

```
{lbxMyListBox.Line(lRow)}
```

Answer

The "LINE" method is associated with the "VALUES" property of the "LISTBOX" control. Here is the corrected code:

```
{lbxMyListBox.Values.Line(lRow)}
```

Question

The following code does not work:

```
{lbxMyListBox.Values.Line({lbxTmp})}
```

Answer

You cannot use a referenced property in a method. Here is what you should write:

```
Dim lRow As Long
lRow = {lbxTmp}
{lbxMyListBox.Values.Line(lRow)}
```

Question

The following code, which assigns a fixed value to a property, does not work:

```
{Property} = 123
```

Answer

To assign a value to a property, you need to use the "AmSetProperty()" function, as shown below:

```
Dim irc as Integer
irc= AmSetProperty("Proprerty", 123)
```

Note:

Don't forget to recover the return code ("irc" in this example), even if you are don't need to use it.

Question

When executing a wizard that creates an asset in the database, the following error message appears:

```
12001 - You do not have write-access rights
```

This message appears even if the user is connected as administrator.

Answer

This message appears when a write-access if attempted outside of the wizard "FINISH.DO" node. The wizard does the following:

1. Collects information via a series of pages (write-access forbidden even for the AssetCenter administrator)
2. Executes the script contained in the "FINISH.DO" node (write-access authorized according to the user's access rights)

Question

Error messages that appear when executing wizard are sometimes incomplete.

Answer

Press SHIFT+F9 to display the debugger. Error messages in the history window are often more explicit.

Question

When the "DBLISTBOX" control is used in a wizard page, performance is impacted. Is this normal?

Answer

This problem occurs when you use the "DBLISTBOX" control in conjunction with a filter. When this is the case, each time the selection changes, a query is sent to the database to make sure that the selection corresponds to the filter. This additional query is not performed when the selection is set by the user.

Question

How do I allow or forbid editing certain columns in the "LISTBOX" control.

Answer

Use the "EDITABLE" property of this control. The value assigned to this property is a string made up of "0" and characters separated by the pipe character "|", which is used a column separator. "0" defines the column as "non-editable", "1" as "editable". If you omit a value, the corresponding column will not be editable, only columns 2 and 4 are editable:

```
EDITABLE = "|1||1"
```

Question

What do I need to do to make a wizard open a detail screen?

Answer

You need to use DDE calls (via a function) inside the wizard. The wizard must not be modal. Here is an example of how to open the table of asset from inside a wizard:

```
Dim irc as Long
irc = AmActionDDE("am", "AssetCenter", "OpenTable(amAsset)")
```

Question

What is the difference between the "COLNAME" and "COLTITLE" properties of a "LISTBOX" control?

Answer

The titles of columns in a "LISTBOX" control can be defined automatically or manually:

- The "COLNAME" property, associated with the "TABLE" property allows you to automatically define the titles of columns in a "LISTBOX" control using field labels from the database.
- The "COLTITLE" property if it is populated forces the titles of the columns. If this property is not defined, the column titles will be those defined by the "COLNAME" property.

Thus the following example:

```
...
TABLE = "amEmplDept"
COLNAME = "Name||FirstName"
COLTITLE = "|A|B"
...
```

displays the following labels in the columns of the "LISTBOX" control: Name, A, B.

The "COLNAME" property also defines the type of control used when the column values are editable.

15 CHAPTER News

This chapter explains how to broadcast and manage news with AssetCenter.

You access the list of news using the **Tools/ News** menu.

You activate/deactivate the news marquee using the **Windows/ Display news marqee** menu or the  in the toolbar.

Definition of a news item

A news item is a piece of information that you want to broadcast to a designated group of persons for a specified period of time.

These persons belong to employee groups.

In general, this type of information is of a short-lived nature.

Example of a news item: "The XXX server will be down for maintenance between 11:00 and 12:00 on 10/02/2002".

Overview of news

Creating a news item

Users with rights in the News table can create news using the **Tools/ News** menu.

In the news detail you will see the:

- Message
- Broadcast list of the message.
- Validity period of the message.

Reading news

The news marquee enables any user to read the messages broadcast to the group of which they are a part.

Importance of news items

To define the importance of a message, you must populate the **Importance** field (SQL: seSeverity) in the news detail.

Each level of importance is associated with a color that you choose in the **Color** field (SQL: IColor), which then appears in the news marquee.

Message to broadcast

The **Message** tab of a news detail contains the text to broadcast.

A message can contain 255 characters.

News broadcast list

The **Broadcast** tab of a news detail lists the employee groups who can read the news item.

All employee groups option (SQL: bAllGroups)

If you select this option, the members of all employee groups will see the message.

Otherwise, only the members of the selected employee groups (in the **Broadcast** tab's list) will see the message.

Use the  ,  and  to add, delete, view or modify the employee groups in the broadcast list.

Include sub-groups option (SLQ bChildGrps)

A message can be broadcast to all employee groups or a selection of groups. If you select this option (which is selected by default) the set of sub-groups of the selected groups will see the message. This is because the employee groups are arranged hierarchically.

Display the news

To view the news:

1 Activate the news marquee.

2 Use the buttons in the news marquee to scroll through the messages.

You can define:

- The colors of how messages are displayed according to their importance.
- The automatic-refresh mode of the news marquee.

Activating the news marquee

Any AssetCenter user can activate the news marquee. There, users will see those messages broadcast to the employee group to which they belong. The news marquee can be activated/deactivated in two ways:

- Using the **Window/Display news marquee** menu.
- Using the  icon in the toolbar.

Table 15.1. News marquee buttons

	Click this button to read new messages before waiting for the news marquee to check for them at its next defined interval (via the **Edit/Options** menu).
	Click this button to display the previous message.
	Click this button to display the next message.
	Click this button to interrupt or continue the scrolling of the news in the marquee.

Indexes

INDEX

A

B

C

D

E

F

G

H

I

P

Q

R

S

T

U

V

W

Y


Peregrine
SYSTEMS